

OPC UA
Test Tool Uses & Analysis Report
Version 1.00 (4/21/2011)

James Wang
USDC, Yokogawa Corporation of America

Editors
Toshio Ono & Paul Hunkar
USDC, Yokogawa Corporation of America

Overview

The OPC UA SDK provided UA Test Application and the OPC UA Compliance Test Tool (CTT) Application can be used to validate a custom OPC UA Server's features. In this report, both tools are used to test the OPC Sample Server. The report includes information regarding the use of the tools, required configuration, test cases, generated results, and the extension of test cases with examples from the test procedures.

Table of Contents

Overview	2
Introduction	4
UA Test Application	6
Software	6
Configuration	6
Test Cases and Results	6
UA Compliance Test Tool (CTT) Application	10
Software	10
Configuration	10
Test Cases and Results	10
Extension of Test Cases	16
Adding a new test case to test project	16
Example 1: Adding a new test case “NewTestCase” in Session test category:	17
Adding a new test case to CTT	18
Example 2: Adding a test case in CTT	19
Example 3: Correct the error reported from an existing CTT test procedure:	21
Example 4: Set debug break points in test procedures	24
Test Application Comments	28
CTT Comments	28
Improvement Recommendation (to OPC)	28
General Usage	31
CTT Server Test Script Programming Notes	32
CTT type libraries	32
Summary	46
Conclusion	47

Introduction

UA Test Application is available from OPC UA SDK download. The Visual Studio project provides a testing environment to consume and validate the specified UA Server Services against a custom UA Server. The following table lists the features for this application:

Feature	Description
Role	UA Client
Build Time IDE	VS 2008 SP1, .Net Framework 3.5, C#, source code available
Run Time IDE	Custom Windows Form. Debug is available in VS 2008
Test Mode	Automated
Test Cases	Fixed in source code, not extendable unless modify source code
Configuration	Minimum
Call Mechanism	Direct, call to UA client library
Performance Info	Execution time of test case is reported
ToDo List	None
Conform with UA profiles specification	No
Iterations	Configurable in repeat count
Sample Count	Configurable in percentage
Security/ Encryption	Configurable in all cases
Log	Reported in test case and end of test case

OPC UA Compliance Test Tool (CTT) Application is the official test program from OPC Foundation for validating the functionality of custom UA Server and client application. It provides the test cases in Conformance Units and groups into Profiles. CTT can run the tests in a set of Profiles or Conformance Unites. In this document, the server test is engaged and reported. The following table lists the features for CTT:

Feature	Description
Role	UA Client
Build Time IDE	Visual Studio C++ application. Application source code is not available.
Run Time IDE	Custom Windows Form application. Test case written in JavaScript. Script source code available. Update/Debug of script at runtime is available.
Test Mode	Automated
Test Cases	Configurable, extendable by adding/update test scripts
Configuration	Very extensive
Call Mechanism	Indirect, call to JavaScript Helper library
Performance Info	Log in Script code
ToDo List	Many test cases are in to-do list

Feature	Description
Conform with UA profiles specification	Yes
Iterations	Fixed
Sample Count	Fixed
Security/ Encryption	Fixed
Log	Comprehensive Script output, log and trace messages

UA Test Application

Software

The software used for the testing includes:

1. Target UA Server – the OPC UA Sample Server.
2. Local Discovery Server (optional) – OPC UA Discovery Service.
3. UA Configuration Tool – used to configure the application certificates.
4. Server Test Tool – the executable of server test tool in UA Test Application (installed from OPC UA SDK 1.01 Test Applications Source Code Setup [321.0 Snapshot].zip)

Configuration

The one requirement prior to start UA Test Application is to configure the application certificate in Windows\Local Machine\UA Applications store and the same for the target UA Server and Local Discovery Server.

Since Server Test Tool runs as a UA Client application to the target UA Server, all configuration requirements are specified in its Client configuration file named Opc.Ua.ServerTestTool.Config.xml. All test cases in Server Test Tool are defined in the <Extensions> tag in the configuration file. It contains <TestCases> and <ServerTestCase> tags. The <ServerTestCase> tag defines the test case including name, the relationship between test cases, etc.

Test Cases and Results

The following table summarizes the test results run by UA test application:

Category	Test Case	UA Service	Implementation	Expected Result
Session	KeepAlive	CreateSession ActivateSession	Change KeepAliveInterval value	KeepAliveCount >= ExpectedKeepAliveCount* *=(testDuration / KeepAliveInterval)
	Reconnect	ITransportChannel.Reconnect BeginActivateSession EndActivateSession	Switch Channel	KeepAliveCount >= ExpectedKeepAliveCount
Browse	Hierarchical	Browse	Find all nodes and children nodes	Collect all nodes in the address space
	ReferenceType	Browse	Set specified ReferenceTypeId IncludeSubtypes BrowseDirection	Collect specified reference nodes
	BrowseResultMask	Browse	Set ResultMask = (uint)BrowseResultMask.None	Collect specified reference nodes

Category	Test Case	UA Service	Implementation	Expected Result
	NodeClass	Browse	Set NodeClass mask	Collect specified reference nodes
	BrowseNext	Browse	Set NodeClass mask & References (Unspecified & NonHierarchicalReferences) and (Object View Method Variable & HierarchicalReferences)	Collect specified reference nodes
Read	Attributes	Read	Read all attributes each node	Read successful for all nodes
	IndexRanges	Read	Set IndexRange = "1:2" and "10000000:20000000"	Read successful for all nodes
	DataEncodings	Read	Set DataEncoding = DefaultBinary and DataEncoding = DefaultXml	Read successful for all nodes
	ReadWrite	Read, Write	Write all attributes for each node	Read/Write successful for all nodes
Write	RandomValues	Write	Select variable w/ CurrentReadOrWrite Write raddom value to Attributes.Value	Write successful for all writable nodes
	TypeMismatch	Write	Write a single value and an array value to mismatch attribute type	Write successful for all writable nodes
Call	MethodTest	Call	Call Methods with BrowseName "MethodTest"	Method call successful
	Errors	Call	Call Methods except with BrowseName "MethodTest"	Method call successful
Translat ePath	SingleHop	TranslateBrowse PathsToNodeIds	Set object RelativePathElement with all ReferenceTypeId and IncludeSubtypes to false	Collect specified nodes and no errors

Category	Test Case	UA Service	Implementation	Expected Result
	MultiHop	TranslateBrowsePathsToNodeIds	Set object RelativePathElement to NonHierarchicalReferences And IncludeSubtypes to true Also to HierarchicalReferences with forward reference Clamp total count to 4	Collect specified nodes and no errors
Subscribe	KeepAlive	CreateSubscription	Set multiple PublishInterval, LifetimeCount and alternating PublishigEnabled	Publish responses received without error.
	PublishingInterval	CreateSubscription ModifySubscription	Set multiple KeepAliveCount Change PublishingInterval	Publish responses received without error.
	CreateItems	CreateMonitoredItems	Set MonitoredItemCreateRequestCollection To all attributes of all nodes	Collect /Verify DataChangeNotification counts Publish responses received without error.
MonitoredItem	SamplingInterval	CreateSubscription CreateMonitoredItems Write	Set multiple MonitoredItem SamplingInterval For each writable variable	Verify MonitoredItem object status sucessful
	ModifySamplingInterval	CreateSubscription CreateMonitoredItems Write	Change MonitoredItem SamplingInterval	Verify MonitoredItem object status sucessful
	QueueSize	CreateSubscription CreateMonitoredItems Write	Set multiple MonitoredItem QueueSize, DiscardOldest	Verify MonitoredItem object status sucessful
	ModifyQueueSize	CreateSubscription CreateMonitoredItems Write	Change MonitoredItem QueueSize	Verify MonitoredItem object status sucessful

Category	Test Case	UA Service	Implementation	Expected Result
	Deadband	CreateSubscription CreateMonitoredItems Write	Set MonitoredItem with DataChangeFilter DeadbandType to Absolute, Percent DeadbandValue to fix value and Trigger to DataChangeTrigger.StatusValue	Verify MonitoredItem object status sucessful
	ModifyDeadband	CreateSubscription CreateMonitoredItems Write	Change MonitoredItem DataChangeFilter. DeadbandValue	Verify MonitoredItem object status sucessful

UA Compliance Test Tool (CTT) Application

Software

The software used for the testing includes:

1. Target UA Server – the OPC UA Sample Server.
2. Local Discovery Server (optional) – OPC UA Discovery Service.
3. UA Configuration Tool – used to configure the application certificates.
4. CTT – the executable program of UA Compliance Test Tool Application (installed from Setup UA Compliance Test Tool V1.0.exe and newer version of Setup UA Compliance Test Tool V1.0.1.exe)

Configuration

CTT is designed to test both OPC UA Clients and Servers. This document covers the testing of a UA Server application. The test tool runs on a project based. A standalone server project is created when a new test for a target UA Server is needed. Open an existing project, unlimited repeated tests can be performed. The following settings are required at runtime to setup the server project before starting testing:

Project Settings	Value
Server URL Discovery URL	Specify the endpoint of the target UA Server
Test Items/Nodes	Assign the requested Node Ids
Sever Test Options	Use default or custom value
Test Tool Options	Use default or custom value
Server Certificates	Copy & assign UA Sever certificates in PKI/CA/certs folder
UA Settings	Use default or custom value
Advanced Setting	Use default or custom value

Test Cases and Results

In the CTT, the Server project collects the test cases defined by two divisions: Profile Categories and Conformance Groups. Profile Categories contain defined UA Profiles. Conformance Groups organize the Conformance Units into defined service sets with a set of test case scripts. The Profile Categories, Profiles and Conformance Groups/Units are defined in file uaprofiles.xml in the test project folder.

CTT Server test cases can be selected by Profiles at runtime are listed in the table below:

Note:

The table only lists non-repeated Conformance Unit in the specific Profile.

Version 1.0.0 and 1.0.1 share the same build version 1.0.0.012 and Plugins V0.1.0.002; however, 1.0.0 has Ua Stack 1.01.310.0 and 1.0.1 has Ua Stack 1.01.320.0.0

Category	Profile	Conformance Unit	Implementation		Test Result	
			1.0.0	1.0.1	1.0.0	1.0.1
Security	SecurityPolicy - Basic128Rsa15	Security Certificate Validation	Done*	Done	√*	√
		Security Basic 128Rsa15	ToDo*	ToDo		
	SecurityPolicy - Basic256	Security Certificate Validation	Done	Done	√	√
		Security Basic 256	ToDo	ToDo		
	SecurityPolicy - None	Security None	ToDo	ToDo		
		Security None CreateSession ActivateSession	ToDo	ToDo		
Server	Auditing Server Facet	Auditing Base	ToDo	ToDo		
	Base Server Behaviour Facet	Protocol Configuration	ToDo	ToDo		
		Security Administration - XML Schema	ToDo	ToDo		
		Security Certificate Administration	ToDo	ToDo		
		Security Administration	ToDo	ToDo		
		Discovery Configuration	ToDo	ToDo		
	Basic DataChange Subscription Server Facet	Monitor Value Change	Done	Done	?	?
		Monitor Items 100	ToDo	ToDo		
		Monitor QueueSize_1	Done	Done	√	√
		Subscription Publish Discard Policy	ToDo	ToDo		
		Monitor Triggering	Done	Done	?*	?
		Monitor Basic	Done	Done	√	√
		Subscription Minimum 02	Done	Done	√	√
		Subscription Publish Min 05	Done	Done	√	√
		Subscription Basic	Done	Done	√	√
	Client Redundancy Facet	Subscription Transfer	Done	Done	?	?
	ComplexType Server Facet	Monitor Alternate Encoding	ToDo	ToDo		
		Attribute Alternate Encoding	ToDo	ToDo		
		Attribute Read Complex	ToDo	ToDo		

Category	Profile	Conformance Unit	Implementation		Test Result	
			1.0.0	1.0.1	1.0.0	1.0.1
		Address Space Complex Datatypes	ToDo	ToDo		
		Attribute Write Complex	ToDo	ToDo		
	Core Server Facet	Security User Name Password	ToDo	ToDo		
		View RegisterNodes	Done	Done	√	√
		Attribute Read	Done	Done	?	?
		Attribute Write Index	Done	Done	?	?
		Security User X509	ToDo	ToDo		
		Session Minimum 10 Parallel	Done	Done	√	√
		Session General Service Behaviour	Done	Done	√	√
		Session Base	Done	Done	√	?
		Discovery Get Endpoints	Done	Done	√	√
		View TranslateBrowsePath	Done	Done	√	√
		Attribute Write Values	Done	Done	√	√
		Security Administration	ToDo	ToDo		
		Discovery Find Servers Self	Done	Done	√	√
		View Minimum Continuation Point 01	Done	Done	?	?
		Base Info Core Structure	ToDo	ToDo		
		Address Space Base	ToDo	ToDo		
		View Basic	Done	Done	?	?
	DataAccess Server Facet	Data Access Semantic Changes	ToDo	ToDo		
		Data Access PercentDeadBand	Done	Done	?	?
		Data Access Analog	Done	Done	?	?
		Data Access DataItems	Done	Done	?	?
		Data Access MultiState	Done	Done	?	?
		Data Access TwoState	Done	Done	?	?
	Embedded UA Server	Base Info Type System	ToDo	ToDo		
	Enhanced DataChange Subscription Server Facet	Monitor MinQueueSize_02	Done	Done	√	√
		Subscription Minimum 05	Done	Done	√	√

Category	Profile	Conformance Unit	Implementation		Test Result	
			1.0.0	1.0.1	1.0.0	1.0.1
		Subscription Publish Min 10	Done	Done	√	√
		Monitor Items 500	ToDo	ToDo		
	Event Subscription Server Facet	Monitor Complex Event Filter	ToDo	ToDo		
		Monitor QueueSize_1	Done	Done	√	√
		Subscription Publish Discard Policy	ToDo			
		Monitor Events	Done	Done	?	?
		Monitor Basic	Done	Done	√	√
		Monitor Items 10	Done	Done	√	√
		Subscription Minimum 02	Done	Done	√	√
		Subscription Publish Min 05	Done	Done	√	√
		Subscription Basic	Done	Done	√	?
	Low End Embedded Device Server	No standalone Test cases				
	Method Server Facet	Method Call	ToDo	ToDo		
		Address Space Method	ToDo	ToDo		
	Node Management Server Facet	Node Management Delete Ref	ToDo	ToDo		
		Node Management Add Ref	ToDo	ToDo		
		Base Info Type System	ToDo	ToDo		
		Base Info Model Change	ToDo	ToDo		
		Node Management Delete Node	ToDo	ToDo		
		Node Management Add Node	ToDo	ToDo		
		Address Space Base	ToDo	ToDo		
	Redundancy Transparent Server Facet	Redundancy Server Transparent	ToDo	ToDo		
	Redundancy Visible Server Facet	Redundancy Server	ToDo	ToDo		
	Standard UA Server	Session Minimum 50 Parallel	Done	Done	√	√
		Base Info Diagnostics	ToDo			
		Attribute Write StatusCode & TimeStamp	Done	Done	√	√
		View Minimum Continuation Point 05	Done	Done	√	√

Category	Profile	Conformance Unit	Implementation		Test Result	
			1.0.0	1.0.1	1.0.0	1.0.1
		View Services	Done	Done	?	?
		Discovery Register	ToDo	ToDo		
Transport	SOAP-HTTP WS-SC UA Binary	Protocol Soap Binary WS Security	ToDo	ToDo		
	SOAP-HTTP WS-SC UA XML	Protocol Soap Xml WS Security	ToDo	ToDo		
	SOAP-HTTP WS-SC UA XML-UA Binary	Protocol Soap Binary WS Security	ToDo	ToDo		
		Protocol Soap Xml WS Security	ToDo	ToDo		
	UA-TCP UA-SC UA Binary	Protocol TCP Binary UA Security	ToDo	ToDo		

Note:

- “ToDo” denotes the script code of the test case was not implemented and it has a placeholder file in Test Cases folder.
- “Done” denotes the script code of the test case is complete and a test result is expected at runtime.
- Test Result “√” mark denotes a complete successful test result after running the test scripts.
- Test Result “?” mark denotes a partial successful test result after running the test scripts.

CTT Server test also can be run by the selection of Conformance Groups/Units or test scripts. The following table lists the Conformance Group and its Conformance Units. The implementation and Result are the same as in above table for each Conformance Unit.

Conformance Group	Conformance Units
Address Space Model	Address Space Complex Datatypes Address Space Method Address Space Base
Attribute Services	Attribute Read Attribute Write Index Attribute Write StatusCode & TimeStamp Attribute Alternate Encoding Attribute Read Complex Attribute Write Values Attribute Write Complex
Auditing	Auditing Base
Base Information	Base Info Diagnostics Base Info Type System Base Info Model Change Base Info Core Structure
Data Access	Data Access Semantic Changes Data Access PercentDeadBand Data Access Analog Data Access DataItems Data Access MultiState Access TwoState
Discovery Services	Discovery Get Endpoints Discovery Find Servers Self Discovery Configuration Discovery Register
Method Services	Method Call

Conformance Group	Conformance Units
Monitored Item Services	Monitor Value Change Monitor MinQueueSize_02 Monitor Complex Event Filter Monitor Items 100 Monitor QueueSize_1 Monitor Alternate Encoding Monitor Triggering Monitor Events Monitor Basic Monitor Items 10 Monitor Items 500
Node Management Services	Node Management Delete Ref Node Management Add Ref Node Management Delete Node Node Management Add Node
Protocol and Encoding	Protocol Configuration Protocol Soap Binary WS Security Protocol TCP Binary UA Security Protocol Soap Xml WS Security
Redundancy	Redundancy Server Transparent Redundancy Server
Security	Security None Security User Name Password Security Administration - XML Schema Security User X509 Security Certificate Validation Security Certificate Administration Security None CreateSession ActivateSession Security Basic 128Rsa15 Security Administration Security Basic 256
Session Services	Session Minimum 50 Parallel Session Minimum 10 Parallel Session General Service Behaviour Session Base Session Cancel
Subscription Services	Subscription Publish Discard Policy Subscription Minimum 05 Subscription Publish Min 10 Subscription Minimum 02 Subscription Transfer Subscription Publish Min 05 Subscription Basic
View Services	View RegisterNodes View Minimum Continuation Point 05 View TranslateBrowsePath View Minimum Continuation Point 01 View Basic

Extension of Test Cases

Adding a new test case to test project

To add a new test case to UA Test Application involves adding entry to the configuration file and test procedure code in the TestBase class in the source file named after the category name. It can be done in the build time:

1. Run Visual Studio 2008 and open UA Test Application. Select UA Server Test Tool.
2. Add the new <ServerTestCase> tag with <Name> to naming the new test case and <Parent> to identify the containing test category in Opc.Ua.ServerTestTool.config.xml file.
3. Open xxxxTest.cs file where xxxx is the name of test category. Edit the public override method Run() in TestBase class to add switch/case for the name of test case and the C# code of test procedure.
4. Use Visual Studio 2008 Debug functions to debug the test procedure.

Example 1: Adding a new test case “NewTestCase” in Session test category:

Edit Opc.Ua.ServerTestTool.Config.xml file. In the <Extensions> <TestCases> tag, add

```
<ServerTestCase>
  <Name>NewTestCase</Name>
  <Parent>Session</Parent>
  <Enabled>true</Enabled>
</ServerTestCase>
```

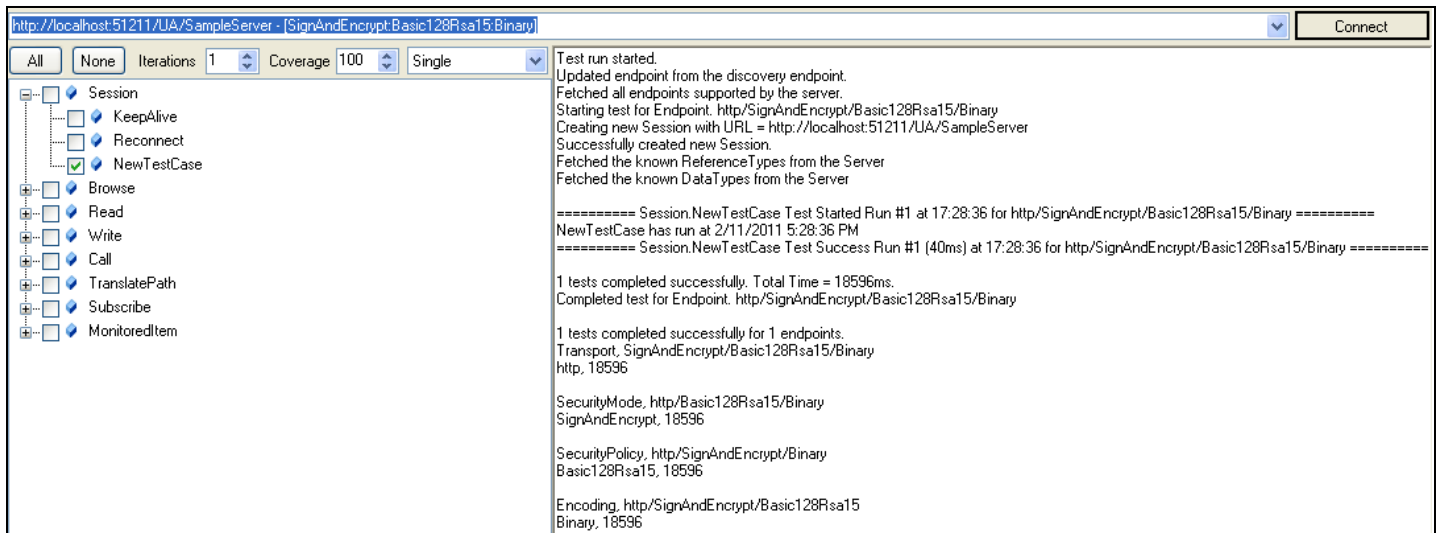
At runtime, the new test case is displayed in the test category pane.



Open SessionTest.cs file. In method Run(), edit the code below:

```
public override bool Run(ServerTestCase testcase, int iteration)
{
    Iteration = iteration
    // do secondary test.
    switch (testcase.Name)
    {
        case "KeepAlive":
        {
            return DoKeepAliveTest();
        }
        case "NewTestCase":
        {
            Log("NewTestCase has run at {0}", DateTime.Now.ToString());
            ReportProgress(MaxProgress);
            return true;
        }
        default:
        case "Reconnect":
        {
            return DoReconnectTest();
        }
    }
}
```

Select Connect to the OPC Sample Server. The test result with the log message is shown below:



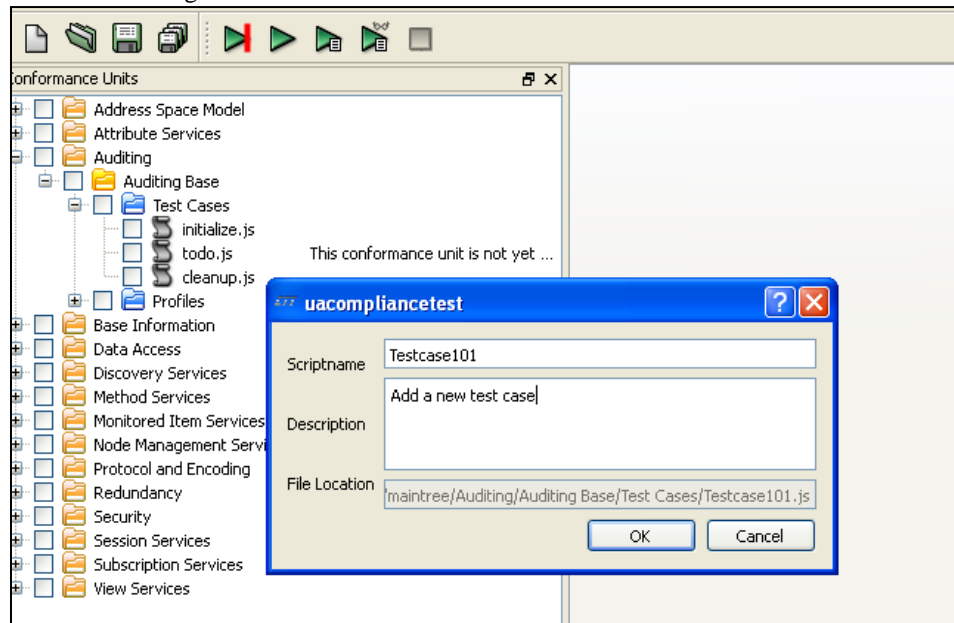
Adding a new test case to CTT

For CTT, Adding a new test case involves adding a new test case in Test Cases container, opening the script working area and editing the source code of test procedure in JavaScript, and debugging and testing the function. However, this process is done in runtime:

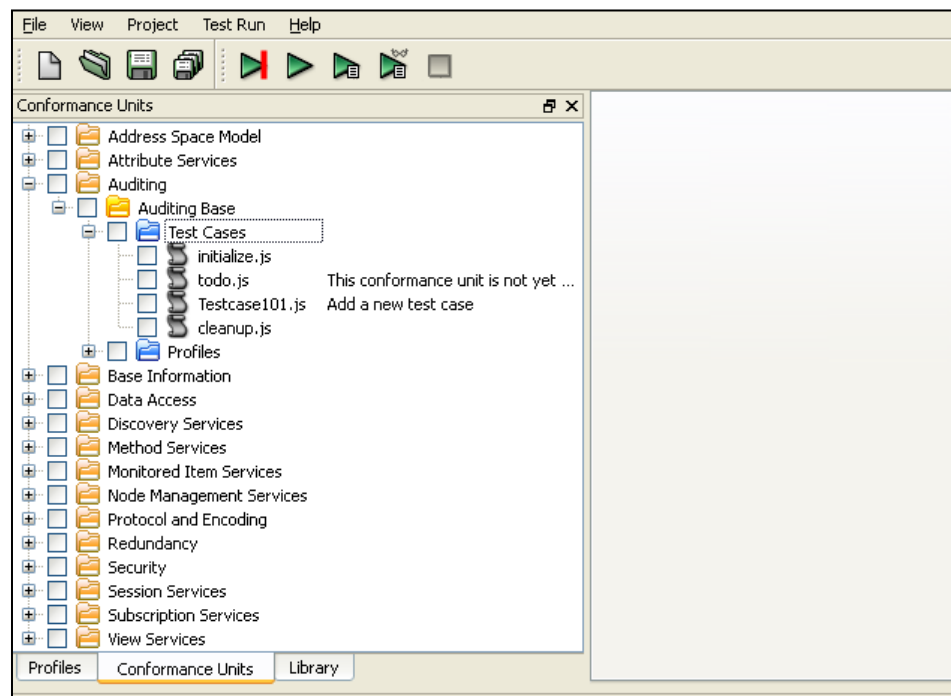
1. Run CTT application and open an existing server project.
2. Select Conformance Units view.
3. Select Test Cases node under the selected Conformance Unit. Right click and select Add Script option.
4. Enter Script Name and Description.
5. Double click the test script node and open the script working area to edit the JavaScript code for test procedure.
6. Use debug buttons to debug and verify the result of test procedure.

Example 2: Adding a test case in CTT

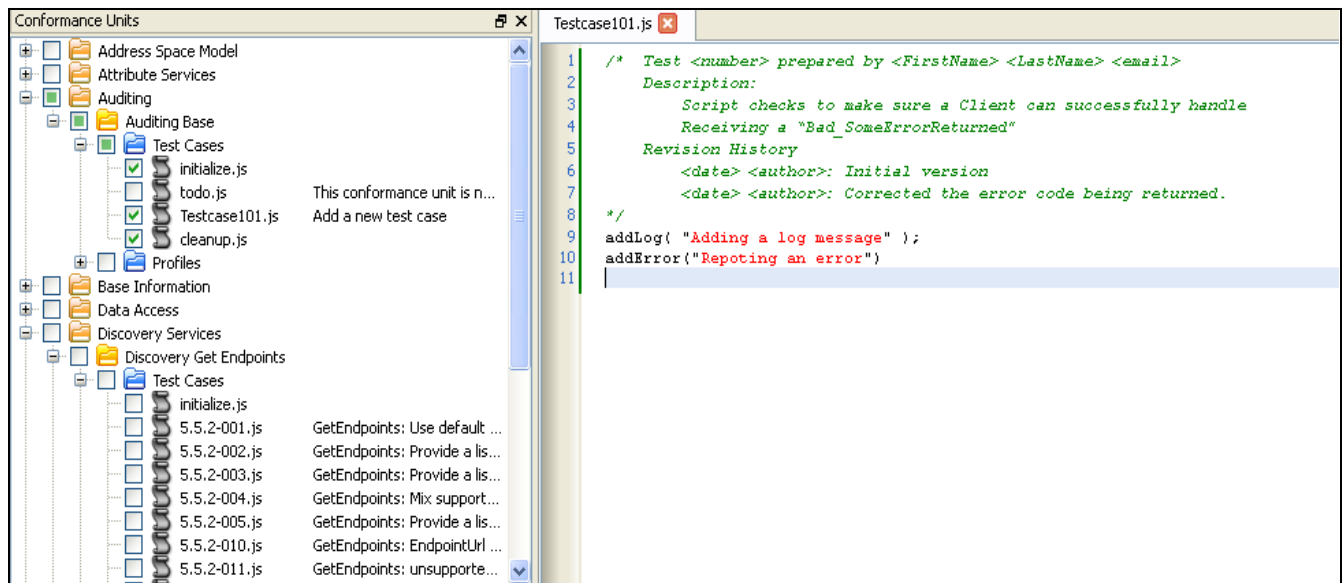
Add a new test case under Auditing Base Conformance Unit:



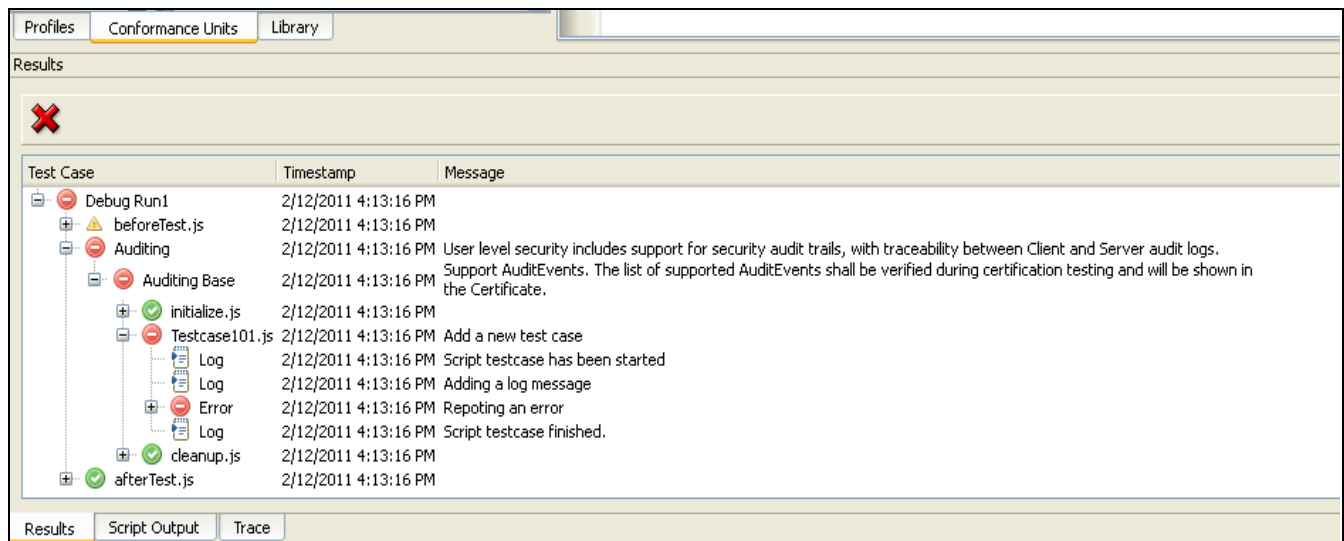
Open the desired location and select Add



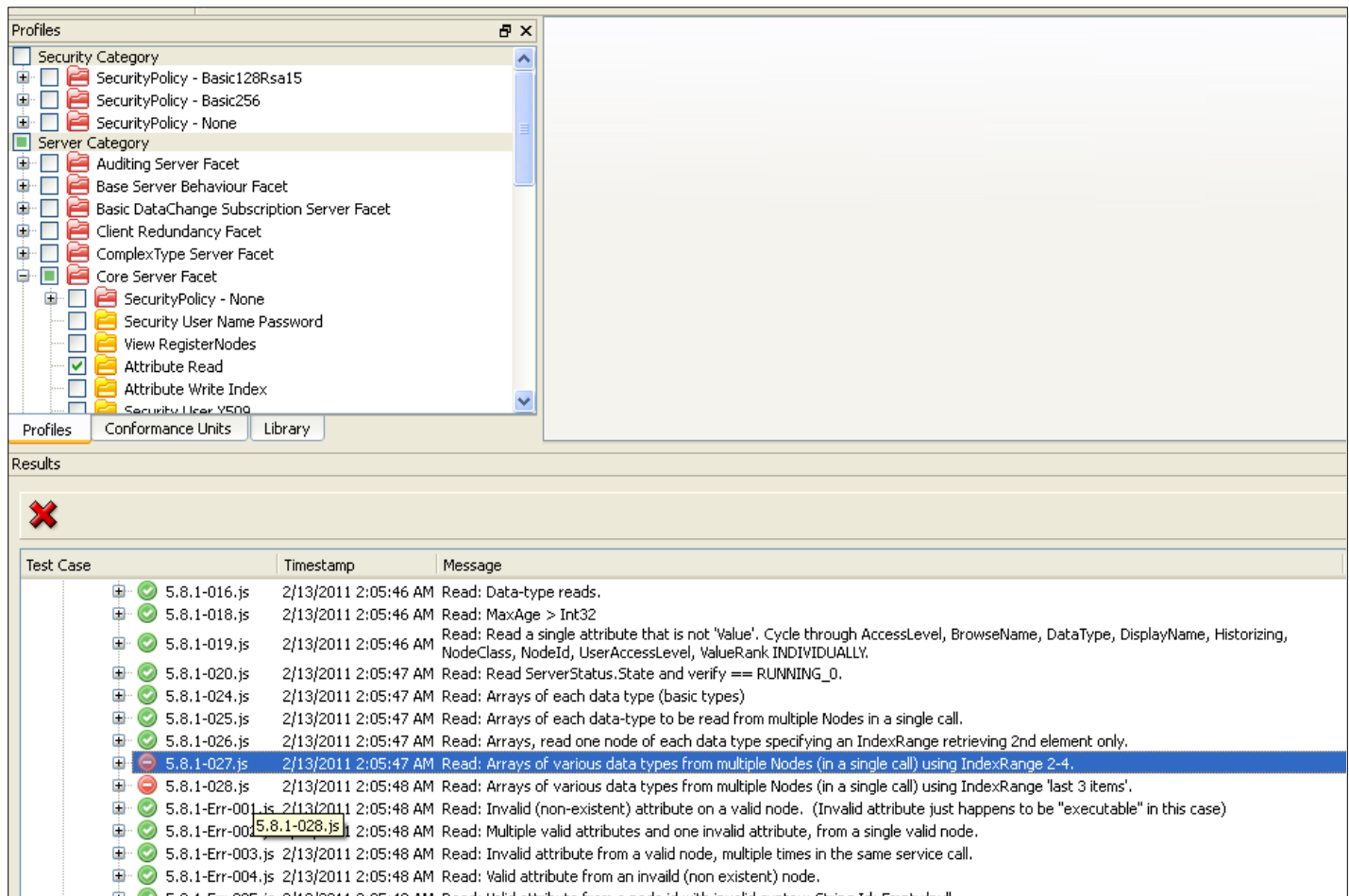
Double click Testcase101.js node and add log message code:



After running the test procedure, the Result pane displays the Log and Error messages:



Example 3: Correct the error reported from an existing CTT test procedure:



The Profiles window shows the following structure:

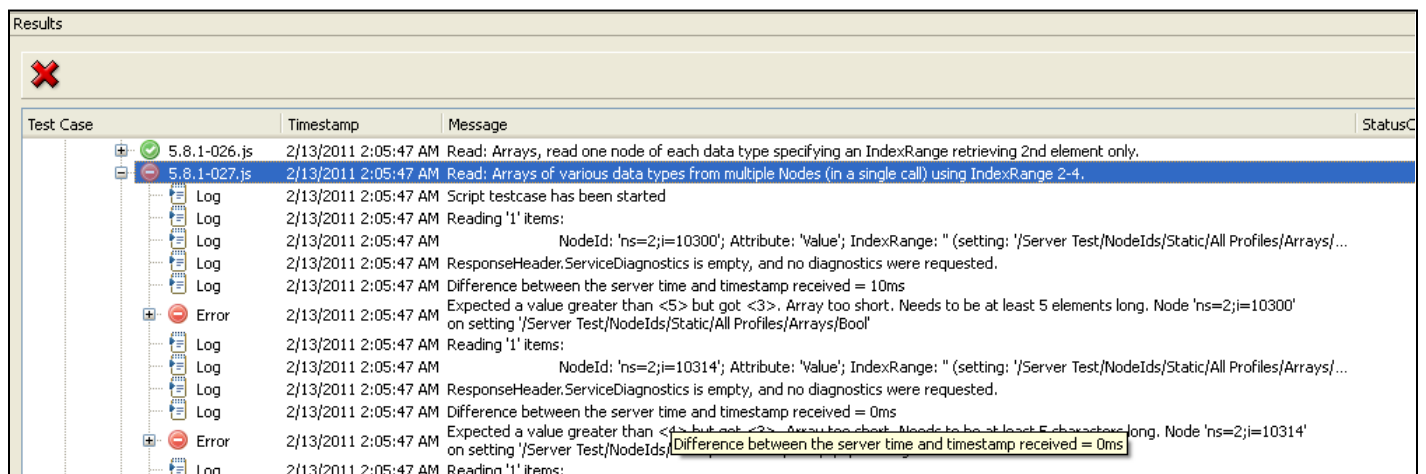
- Security Category
 - SecurityPolicy - Basic128Rsa15
 - SecurityPolicy - Basic256
 - SecurityPolicy - None
- Server Category
 - Auditing Server Facet
 - Base Server Behaviour Facet
 - Basic DataChange Subscription Server Facet
 - Client Redundancy Facet
 - ComplexType Server Facet
 - Core Server Facet
 - SecurityPolicy - None
 - Security User Name Password
 - View RegisterNodes
 - Attribute Read
 - Attribute Write Index
 - Security User Y500

The Results window shows the following test cases:

Test Case	Timestamp	Message
5.8.1-016.js	2/13/2011 2:05:46 AM	Read: Data-type reads.
5.8.1-018.js	2/13/2011 2:05:46 AM	Read: MaxAge > Int32
5.8.1-019.js	2/13/2011 2:05:46 AM	Read: Read a single attribute that is not 'Value'. Cycle through AccessLevel, BrowseName, DataType, DisplayName, Historizing, NodeClass, NodeId, UserAccessLevel, ValueRank INDIVIDUALLY.
5.8.1-020.js	2/13/2011 2:05:47 AM	Read: Read ServerStatus.State and verify == RUNNING_0.
5.8.1-024.js	2/13/2011 2:05:47 AM	Read: Arrays of each data type (basic types)
5.8.1-025.js	2/13/2011 2:05:47 AM	Read: Arrays of each data-type to be read from multiple Nodes in a single call.
5.8.1-026.js	2/13/2011 2:05:47 AM	Read: Arrays, read one node of each data type specifying an IndexRange retrieving 2nd element only.
5.8.1-027.js	2/13/2011 2:05:47 AM	Read: Arrays of various data types from multiple Nodes (in a single call) using IndexRange 2-4.
5.8.1-028.js	2/13/2011 2:05:48 AM	Read: Arrays of various data types from multiple Nodes (in a single call) using IndexRange 'last 3 items'.
5.8.1-Err-001.js	2/13/2011 2:05:48 AM	Read: Invalid (non-existent) attribute on a valid node. (Invalid attribute just happens to be "executable" in this case)
5.8.1-Err-002.js	2/13/2011 2:05:48 AM	Read: Multiple valid attributes and one invalid attribute, from a single valid node.
5.8.1-Err-003.js	2/13/2011 2:05:48 AM	Read: Invalid attribute from a valid node, multiple times in the same service call.
5.8.1-Err-004.js	2/13/2011 2:05:48 AM	Read: Valid attribute from an invalid (non-existent) node.
5.8.1-Err-005.js	2/13/2011 2:05:48 AM	Read: Valid attribute from a node id with invalid context. String Id: Empty/Null

Run the Attribute Read Conformance Unit in Core Server Facet profile. An error in test case 5.8.1-027.js is reported:

Expand the log messages in 5.8.1-027.js, a list of detail error log messages are displayed. It indicated the array dimension was expected for greater than 5.



The Results window shows the following log messages for test case 5.8.1-027.js:

Test Case	Timestamp	Message	Status
5.8.1-026.js	2/13/2011 2:05:47 AM	Read: Arrays, read one node of each data type specifying an IndexRange retrieving 2nd element only.	Pass
5.8.1-027.js	2/13/2011 2:05:47 AM	Read: Arrays of various data types from multiple Nodes (in a single call) using IndexRange 2-4.	Fail
Log	2/13/2011 2:05:47 AM	Script testcase has been started	
Log	2/13/2011 2:05:47 AM	Reading '1' items:	
Log	2/13/2011 2:05:47 AM	NodeId: 'ns=2;i=10300'; Attribute: 'Value'; IndexRange: " (setting: '/Server Test/NodeIds/Static/All Profiles/Arrays/...)	
Log	2/13/2011 2:05:47 AM	ResponseHeader.ServiceDiagnostics is empty, and no diagnostics were requested.	
Log	2/13/2011 2:05:47 AM	Difference between the server time and timestamp received = 10ms	
Error	2/13/2011 2:05:47 AM	Expected a value greater than <5> but got <3>. Array too short. Needs to be at least 5 elements long. Node 'ns=2;i=10300' on setting '/Server Test/NodeIds/Static/All Profiles/Arrays/Bool'	Fail
Log	2/13/2011 2:05:47 AM	Reading '1' items:	
Log	2/13/2011 2:05:47 AM	NodeId: 'ns=2;i=10314'; Attribute: 'Value'; IndexRange: " (setting: '/Server Test/NodeIds/Static/All Profiles/Arrays/...)	
Log	2/13/2011 2:05:47 AM	ResponseHeader.ServiceDiagnostics is empty, and no diagnostics were requested.	
Log	2/13/2011 2:05:47 AM	Difference between the server time and timestamp received = 0ms	
Error	2/13/2011 2:05:47 AM	Expected a value greater than <5> but got <3>. Array too short. Needs to be at least 5 elements long. Node 'ns=2;i=10314' on setting '/Server Test/NodeIds/Static/All Profiles/Arrays/Bool'	Fail
Log	2/13/2011 2:05:47 AM	Reading '1' items:	

Check the setting for the test nodes. Node Id Bool (ns=2;i=10300) has array size 3:

Server Test	
NodeIds	
Static	
All Profiles	
Scalar	
Arrays	
Scalar Set 1	
DA Profile	
Dynamic	
References	
Paths	
Events	
NodeClasses	
Session	
Test Tool	
Ua Settings	
Advanced	

Bool	ns=2;i=10300
ByteString	ns=2;i=10314
DateTime	ns=2;i=10312
Double	ns=2;i=10310
Guid	ns=2;i=10313
Int16	ns=2;i=10303
Int32	ns=2;i=10305
String	ns=2;i=10311
UInt16	ns=2;i=10304
UInt32	ns=2;i=10306
UInt64	ns=2;i=10308
Float	ns=2;i=10309
XmlElement	ns=2;i=10315
Int64	ns=2;i=10307

Bool	ns=2;i=10300
ByteString	ns=2;i=10314
DateTime	ns=2;i=10312
Double	ns=2;i=10310
Guid	ns=2;i=10313
Int16	ns=2;i=10303
Int32	ns=2;i=10305
String	ns=2;i=10311
UInt16	ns=2;i=10304
UInt32	ns=2;i=10306
UInt64	ns=2;i=10308
Float	ns=2;i=10309
XmlElement	ns=2;i=10315
Int64	ns=2;i=10307

Browser Connected
You can drag nodes from the browser and drop them into a settings field

Data

- Conditions
- Dynamic
- Static
- AnalogArray
- AnalogScalar
- Array
- BooleanValue
- ByteStringValue
- ByteValue

Attributes

Attribute	Value
NodeId	ns=2;i=10300
NodeClass	Variable
BrowseName	NamespaceIndex: 2, Name: BooleanW...
DisplayName	Locale: , Text: BooleanValue
Description	Null
WriteMask	0
UserWriteMask	0
Value	Size: 3 [0] = false [1] = true [2] = fal...
DataType	Boolean NodeId: i=1

Check BooleanValue of User Array. It has size 43. Other nodes also have array size more than 5:

Bool	ns=2;i=10300
ByteString	ns=2;i=10314
DateTime	ns=2;i=10312
Double	ns=2;i=10310
Guid	ns=2;i=10313
Int16	ns=2;i=10303
Int32	ns=2;i=10305
String	ns=2;i=10311
UInt16	ns=2;i=10304
UInt32	ns=2;i=10306
UInt64	ns=2;i=10308
Float	ns=2;i=10309
XmlElement	ns=2;i=10315
Int64	ns=2;i=10307

Browser Connected
You can drag nodes from the browser and drop them into a settings field

- AnalogArray
- AnalogScalar
- Array
- MethodTest
- Scalar
- UserArray
- BooleanValue
- ByteStringValue
- ByteValue

Attributes

Attribute	Value
NodeId	ns=2;i=10463
NodeClass	Variable
BrowseName	NamespaceIndex: 2, Name: BooleanV...
DisplayName	Locale: , Text: BooleanValue
Description	Null
WriteMask	0
UserWriteMask	0
Value	Size: 43 [0] = true [1] = true [2] = tr...
DataType	Boolean NodeId: ns=2;i=9898

Edit NodeId setting from UserArray instead of Array set:

Type	NodeId
Bool	ns=2;i=10463
ByteString	ns=2;i=10477
DateTime	ns=2;i=10475
Double	ns=2;i=10473
Guid	ns=2;i=10476
Int16	ns=2;i=10466
Int32	ns=2;i=10468
String	ns=2;i=10474
UInt16	ns=2;i=10467
UInt32	ns=2;i=10469
UInt64	ns=2;i=10471
Float	ns=2;i=10472
XmlElement	ns=2;i=10471
Int64	ns=2;i=10470

The NodeId of a Node where the Value attribute is of Type: BOOL[]
Access: Read/Write
Preferably a Node that supports writes with an IndexRange, if supported.

Re-run the test case, the error is fixed and passed:

Profiles

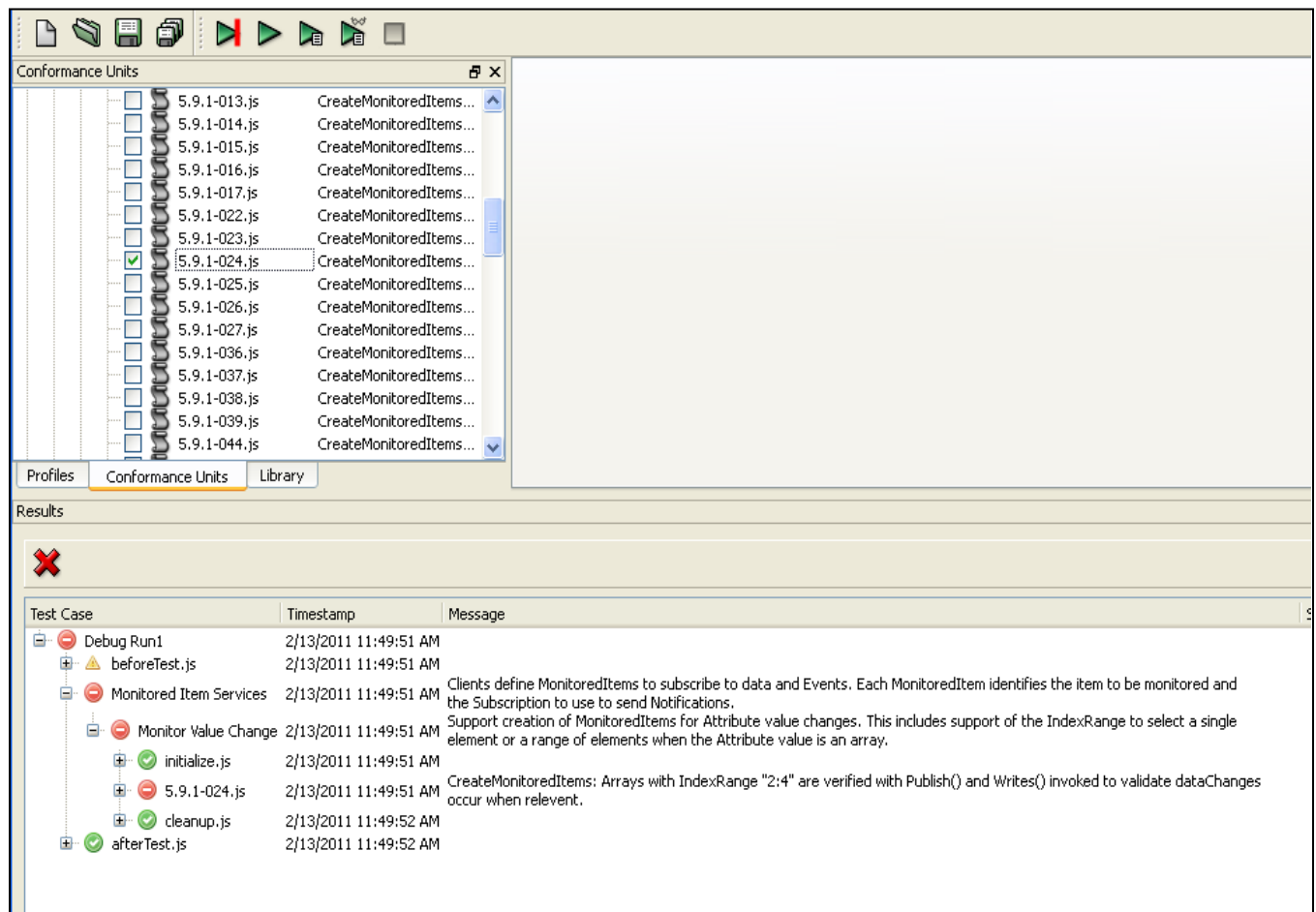
- Security Category
 - SecurityPolicy - Basic128Rsa15
 - SecurityPolicy - Basic256
 - SecurityPolicy - None
- Server Category
 - Auditing Server Facet
 - Base Server Behaviour Facet
 - Basic DataChange Subscription Server Facet
 - Client Redundancy Facet
 - ComplexType Server Facet
 - Core Server Facet
 - SecurityPolicy - None
 - Security User Name Password
 - View RegisterNodes
 - Attribute Read
 - Attribute Write Index
 - Security User V500

Results

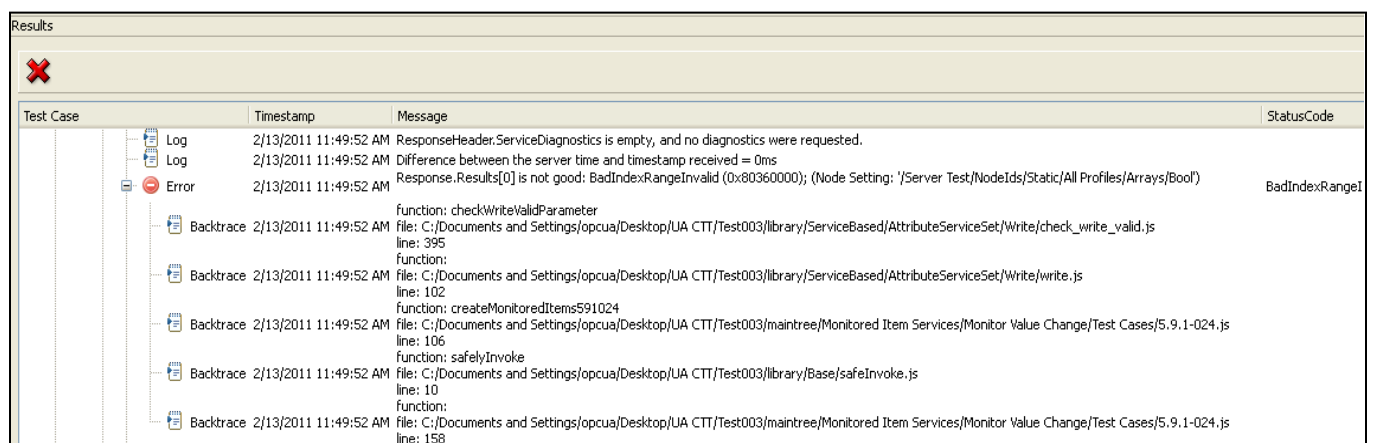
Test Case	Timestamp	Message	StatusCode
5.8.1-009.js	2/13/2011 10:17:02 AM	Read a data value with timestampsToReturn = NEITHER	
5.8.1-010.js	2/13/2011 10:17:02 AM	Read: BrowseName attribute from a valid node.	
5.8.1-011.js	2/13/2011 10:17:02 AM	Read: BrowseName attribute from multiple valid nodes.	
5.8.1-012.js	2/13/2011 10:17:02 AM	Read all available attributes from multiple valid nodes of each different node type, in a single call.	
5.8.1-013.js	2/13/2011 10:17:02 AM	Read: Same attribute from same node multiple times.	
5.8.1-016.js	2/13/2011 10:17:02 AM	Read: Data-type reads.	
5.8.1-018.js	2/13/2011 10:17:03 AM	Read: MaxAge > Int32	
5.8.1-019.js	2/13/2011 10:17:03 AM	Read: Read a single attribute that is not 'Value'. Cycle through AccessLevel, BrowseName, DataType, DisplayName, Historizing, NodeClass, NodeId, UserAccessLevel, ValueRank INDIVIDUALLY.	
5.8.1-020.js	2/13/2011 10:17:03 AM	Read: Read ServerStatus.State and verify == RUNNING_0.	
5.8.1-024.js	2/13/2011 10:17:03 AM	Read: Arrays of each data type (basic types)	
5.8.1-025.js	2/13/2011 10:17:03 AM	Read: Arrays of each data-type to be read from multiple Nodes in a single call.	
5.8.1-026.js	2/13/2011 10:17:03 AM	Read: Arrays, read one node of each data type specifying an IndexRange retrieving 2nd element only.	
5.8.1-027.js	2/13/2011 10:17:03 AM	Read: Arrays of various data types from multiple Nodes (in a single call) using IndexRange 2-4.	
5.8.1-028.js	2/13/2011 10:17:04 AM	Read: Arrays of various data types from multiple Nodes (in a single call) using IndexRange 'last 3 items'.	

Example 4: Set debug break points in test procedures

This example demonstrates how to set the break points in the test script and its sub-scripts. The break points can only be set after issuing the command “Start Current Script in Debugger”. They can not be saved for repeat tests. The break points in the sub-script can be only set after “Step Into” command is issued at the break point in the parent script. The relations and call trace list can be found in the Result pane. This example tested the script 5.9.1-024.js. An error was reported in CreateMonitoredItems Conformance Unit.



The call trace list was shown when expand the error message node:

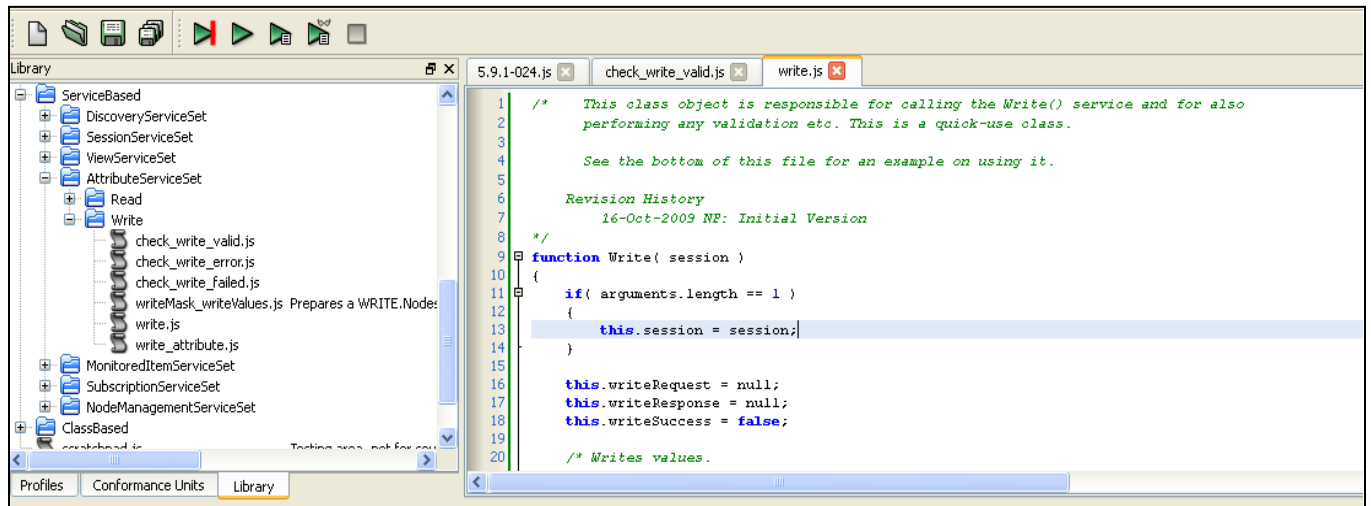



```

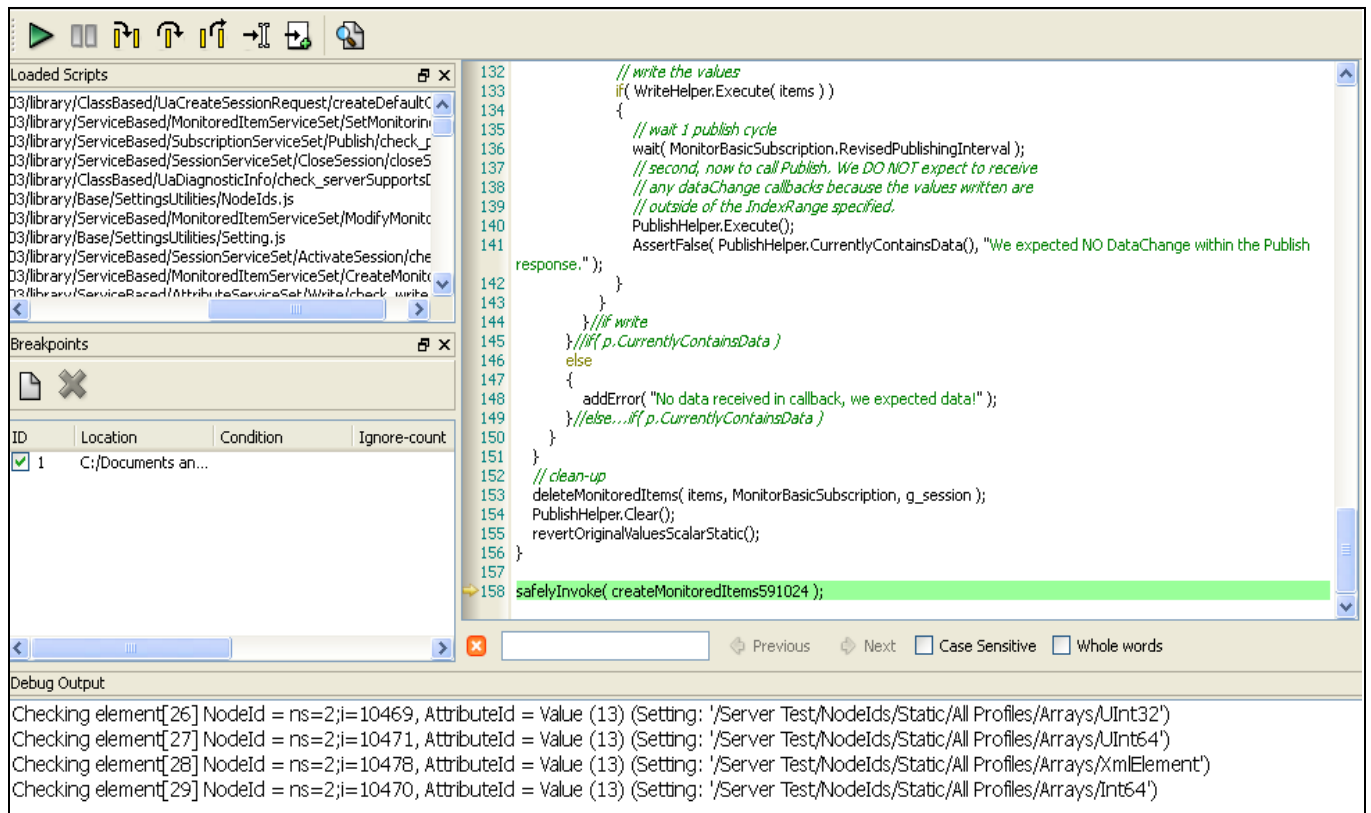
function: checkWriteValidParameter
file: C:/Documents and Settings/opcua/Desktop/UA CTT/Test003/library/ServiceBased/AttributeServiceSet/Write/check_write_valid.js
line: 395
function:
file: C:/Documents and Settings/opcua/Desktop/UA CTT/Test003/library/ServiceBased/AttributeServiceSet/Write/write.js
line: 102
function: createMonitoredItems591024
file: C:/Documents and Settings/opcua/Desktop/UA CTT/Test003/maintree/Monitored Item Services/Monitor Value Change/Test Cases/5.9.1-024.js
line: 106
function: safelyInvoke
file: C:/Documents and Settings/opcua/Desktop/UA CTT/Test003/library/Base/safeInvoke.js
line: 10
function:
file: C:/Documents and Settings/opcua/Desktop/UA CTT/Test003/maintree/Monitored Item Services/Monitor Value Change/Test Cases/5.9.1-024.js
line: 158

```

Load the source of the test script, help script and library service script according to the call trace list:



Select 5.9.1-24.js and Click the button “Start the current script in debugger”:



The screenshot shows the Visual Studio IDE with the following components:

- Loaded Scripts:** A list of loaded scripts on the left, including '03/library/ClassBased/UaCreateSessionRequest/createDefaultC...', '03/library/ServiceBased/MonitoredItemServiceSet/SetMonitori...', '03/library/ServiceBased/SubscriptionServiceSet/Publish/check_...', '03/library/ServiceBased/SessionServiceSet/CloseSession/closeS...', '03/library/ClassBased/UaDiagnosticInfo/check_serverSupportSt...', '03/library/Base/SettingsUtilities/NodeIds.js', '03/library/ServiceBased/MonitoredItemServiceSet/ModifyMonit...', '03/library/Base/SettingsUtilities/Setting.js', '03/library/ServiceBased/SessionServiceSet/ActivateSession/che...', '03/library/ServiceBased/MonitoredItemServiceSet/CreateMonit...', and '03/library/ServiceBased/AttributeServiceSet/WriteCheck_writa...'.
- Breakpoints:** A pane below the loaded scripts showing a single breakpoint at 'C:/Documents an...'. The table has columns for ID, Location, Condition, and Ignore-count.
- Code Editor:** The main editor window displays the 'PublishHelper.js' file. The code includes comments in green and JavaScript code in black. The code is as follows:


```

87     case BuiltInType.ByteString: break;
88     case BuiltInType.String: break;
89     default:
90         addError( "Type: " + BuiltInType.toString( currentType ) + "; Non array value received: " +
91             currentDataChange.MonitoredItems[m].Value.toString() +
92             "; ClientHandle: " + currentDataChange.MonitoredItems[m].ClientHandle );
93         break;
94     }
95     }
96     }
97     }
98     }
99     }
100     PublishHelper.SetMonitoredItemTypesFromDataChange( items );
101
102     // Now, WRITE to the items
103     // first, let's create set some values within the MonitoredItems
104     SetArrayWriteValuesInMonitoredItems( items, 3, 24 );
105
106     if( WriteHelper.Execute( items ) )
107     {
108         // we'll check the first result because if its bad, then the remainder will
109         // likely be bad too...
110         if( WriteHelper.writeResponse.Results[0].isGood() )
111         {
112             // wait 1 publish cycle
113             wait( MonitorBasicSubscription.RevisedPublishingInterval );
114             // verify that a DataChangeNotification occurred after writing

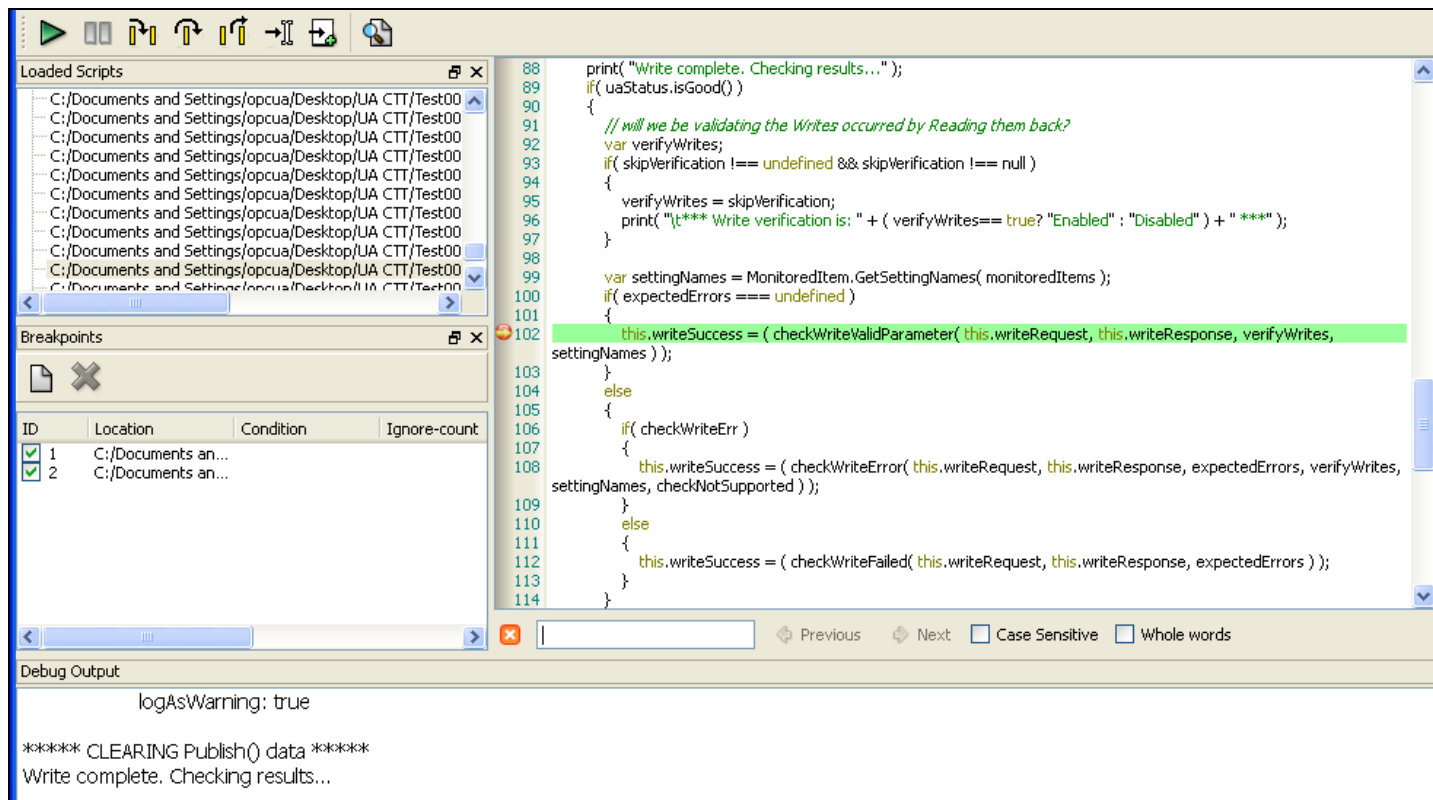
```
- Debug Output:** The bottom pane shows the execution of the code, displaying the results of the 'publish()' function. The output is as follows:


```

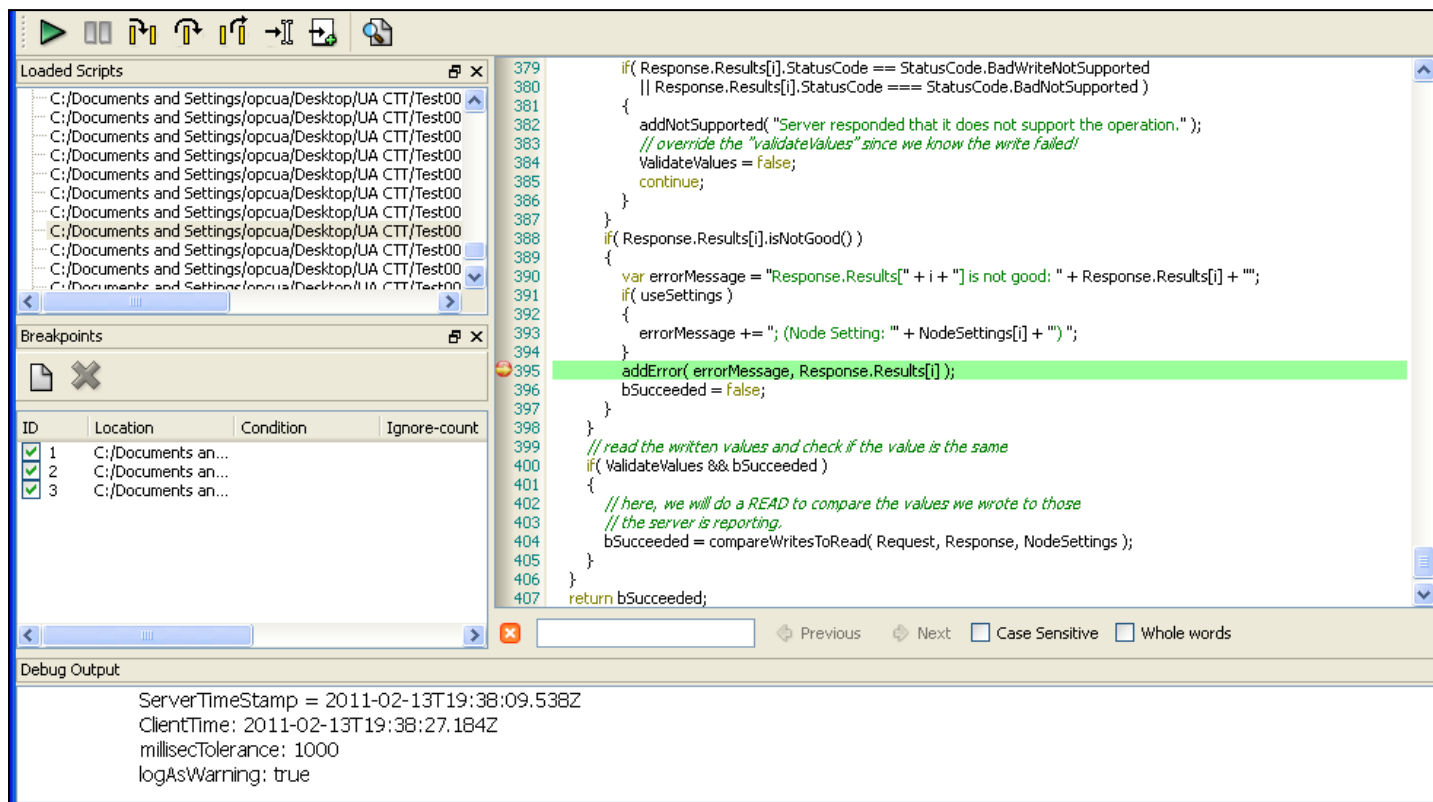
Checking element[26] NodeId = ns=2;i=10469, AttributeId = Value (13) (Setting: '/Server Test/NodeIds/Static/All Profiles/Arrays/UInt32')
Checking element[27] NodeId = ns=2;i=10471, AttributeId = Value (13) (Setting: '/Server Test/NodeIds/Static/All Profiles/Arrays/UInt64')
Checking element[28] NodeId = ns=2;i=10478, AttributeId = Value (13) (Setting: '/Server Test/NodeIds/Static/All Profiles/Arrays/XmlElement')
Checking element[29] NodeId = ns=2;i=10470, AttributeId = Value (13) (Setting: '/Server Test/NodeIds/Static/All Profiles/Arrays/Int64')

```





Step Into line 102 and set break point at line 395 where the error message is reported:



After viewing the Response result, the BadIndexRangeInvalid error was returned from Write service in AttributeServiceSet library. It is indeed that the index range write was not supported in OPC Sample Server.

Test Application Comments

The Test Application can be utilized in two manners by developers. It can be used as additional examples of client code. It can also be utilized as a test application to run against a server. To expand to enhance test cases requires some knowledge regarding OPC UA coding and as such it would not be ideal to use as a test bed for new information model functionality. The lack of a formal report or summary type report also makes it less desirable as a test lab tool. In general the Test Application is acceptable as a tool to be used by OPC UA developers.

CTT Comments

Improvement Recommendation (to OPC)

While reviewing the OPC UA CTT tool numerous issues and improvement ideas and suggestion came up. These ideas are group according to who would benefit from the updates. It would be recommended that the following item be corrected or improved in the CTT tool for it to be more useful as a formal test tool:

- The version information that is displayed by the CTT should include version of framework (including a build), an overall version and any versions for included software.
- The CTT may want to consider including version information for test cases. It may be enough to version test script libraries, but some manner of tracking the test scripts that are being run would be required.
- Profile view does not show test cases, but conformance unit screen does, most people would be interested in the profile view, especially since results are profile based. This view should also be able to be expanded to display actual test cases.
- Adding test cases is useful, but is there a mechanism to ensure that OPC Test cases are run and that a user has not altered the test case? The CTT needs to be able to be run in a Formal Test Mode where the test case that are executed are not the Java script but a compiled or binary version of the scripts that cannot be altered in any manner.
- On first install and opening of CTT the default location is libraries tab – it should open on the profiles tab (It was different between 1.0.0 and 1.0.1). This would be the Tab that a formal test would be run from.
- In general the configuration of the OPC UA CTT is difficult. It requires finding and configuring many tags or options. The configuration must be simplified and improved. The following item offer some suggestion or particular items that could be improved to make the configuration more usable. A major suggestion would be to work more like the Test Application in that it would automatically browse for all required items (maybe a check box allow a choice of automatic vs manual configuration). Any required Items that are missing would be flagged.
- The configuration screen provide a tip that appears to be what the tag would be in the sample server for the given test, but the paths the configuration screen provides as a tip do not match what is actually the sample server and in some case the test still fails. It would be much more helpful if it would just find a tag that is of the appropriate type or if a user selects a tag that a

quick check is made during the configuration validating that the tag is of the correct type and has all the features required for the test.

- The test tool allows a user to double click on a script error and it opens the script which is great. It would be nice if the configuration screen would allow a user to double click on an item (configured nodeid) and it would automatically call up details on the node or move the browse window to the item. In general, it would be helpful to be able to find what the node is that is configured (more information than just a nodeid).
- The tool should also provide some manner to generate a result that could be uploaded to the OPC Foundation or in some manner used to report the compliance of the product, much like the existing OPC Classic CTT does. This result file must include detailed information regard the OPC UA Server or Client that can be used to identify the product that was tested.
- Being able to output the test run results to a file (XML or CSV format) will be helpful for comparing the test results or tracking the changes of test procedure. Currently, it is a tree view control. The content can't be copied.
- Ensure that test cases are provided for all functionality that is exposed by a standard server or any one of the standalone Profiles, not just the functionality of some of the facets that require additional functionality to make a complete server. Test cases may be classified as Required or Optional to indicate its importance.
- The tool needs to be able to test all of the supported transport and security facets. It is understood that some platforms may not support all of these options, but if any platform supports them (i.e. windows) then they must be included in the CTT.

The following updates should be considered to improve the usability of the CTT tool as a development Verification/test tool:

- The CTT tool needs to be able to save debug information as part of a project and automatically reload break points for all scripts when the project is loaded. Setting break points is very difficult; they have to be set as you load a java script, this can be very time consuming. The tool needs to be able to load and save all breakpoint in a java script as part of a project. Making a change to the java script could reset break point, but only if the test script code changes significantly.
- Error response receive back in many cases are very hard to understand. They make no sense and appear out of context, this seems to be especially true with errors raised by library functions. Library functions should return error information to the calling function and let the calling function generate a more appropriate error test message. Currently to understand many of the errors a user has to debug the test scripts and actually either set a breakpoint in the appropriate library function or at least open the library and look at the code to understand what the error is. This sort of work does not assist a server developer in debugging their own code or for a tester to determine any configuration errors.
- It would also be nice to allow a SaveAs option to dump a project to a different name, i.e. create a new instance of a project etc. This feature could be used to create an archive of a project, say when pass one of testing is complete. Any new runs of the project would not affect this archived copy, which could be used as a reference. It could also be used to create new version of a

project for debugging new problems, keeping the same set of break points already loaded. If custom test cases are developed it would allow an easier method of saving them for a new run.

The following updates should be considered to improve the usability of the CTT tool with regard to usage as an test bed for additional OPC UA custom information model test cases:

- Test cases that are added are only added to a project, There is no way to copy them to be global for all projects or all new projects. It would be nice to be able to generate global test cases that can be included in a new project.
- It would be nice if a manner existed for marking a library of functions as global, so that it would automatically be included in all future projects or allow it to be easily added to a project.
- User defined test cases must be added to a separate folder, under an existing conformance unit. They should not be allowed to be intermixed with OPC Foundation provided test scripts. This would apply when a vendor want define additional test case that apply to their particular server. If a user is using the CTT to develop test scripts that are to be part of the OPC Foundation UA CTT, then a custom (non-release version) of the tool must be used. This version would not be allowed to be used to generate formal test results.
- Results of a run cannot be saved in any manner that we can see, i.e. no cut and paste or other manner to dump results to a report. Very often a report would need to be generated documenting the run and results. It would be nice to be able to save the results in some manner other than a screen copy.

General Usage

Currently the OPC UA CTT appears to be geared toward a developer developing CTT test cases, not a formal CTT tool or even a tool for a developer to use to test their OPC UA server/client. This does not mean that it could not be used by a developer to test an OPC UA server or client or with some enhancements as a formal test tool.

Many of the features provided by the CTT are very useful features and make it appealing both as a developer tool and as a tool for formal testing or a tool for use in custom Yokogawa information model testing. The ability to define new test cases for information model items without having to worry about the infrastructure associated with OPC UA is particularly helpful. The requested improvements would greatly enhance the usability of the product for development and formal testing, but even without the improvement it could be used.

It is worth noting that the version of the tool that was tested is still not a released version, and that it is assumed that a release version will more completely cover the many missing test cases (those listed as ToDo). This would be especially true if the product is to be used as a formal CTT test for indicating compliance of a server or client to the OPC UA standard. To list a product as Compliant the tool must be able to test all of the functionality a server or client exposes. If it cannot the indication should not be Compliant at the product level, maybe only an indication of the Profile for which it was compliant and an indication of the functionality that was not tested.

CTT Server Test Script Programming Notes

This section provides helpful notes to the developer of test scripts. This document only describes commonly used scripts and the general use of scripts. The summary table contains a list of name and cross reference of the available library scripts. It is expected that this document will grow as additional library items are described or as the list of library function provided by the OPC foundation is enhanced.

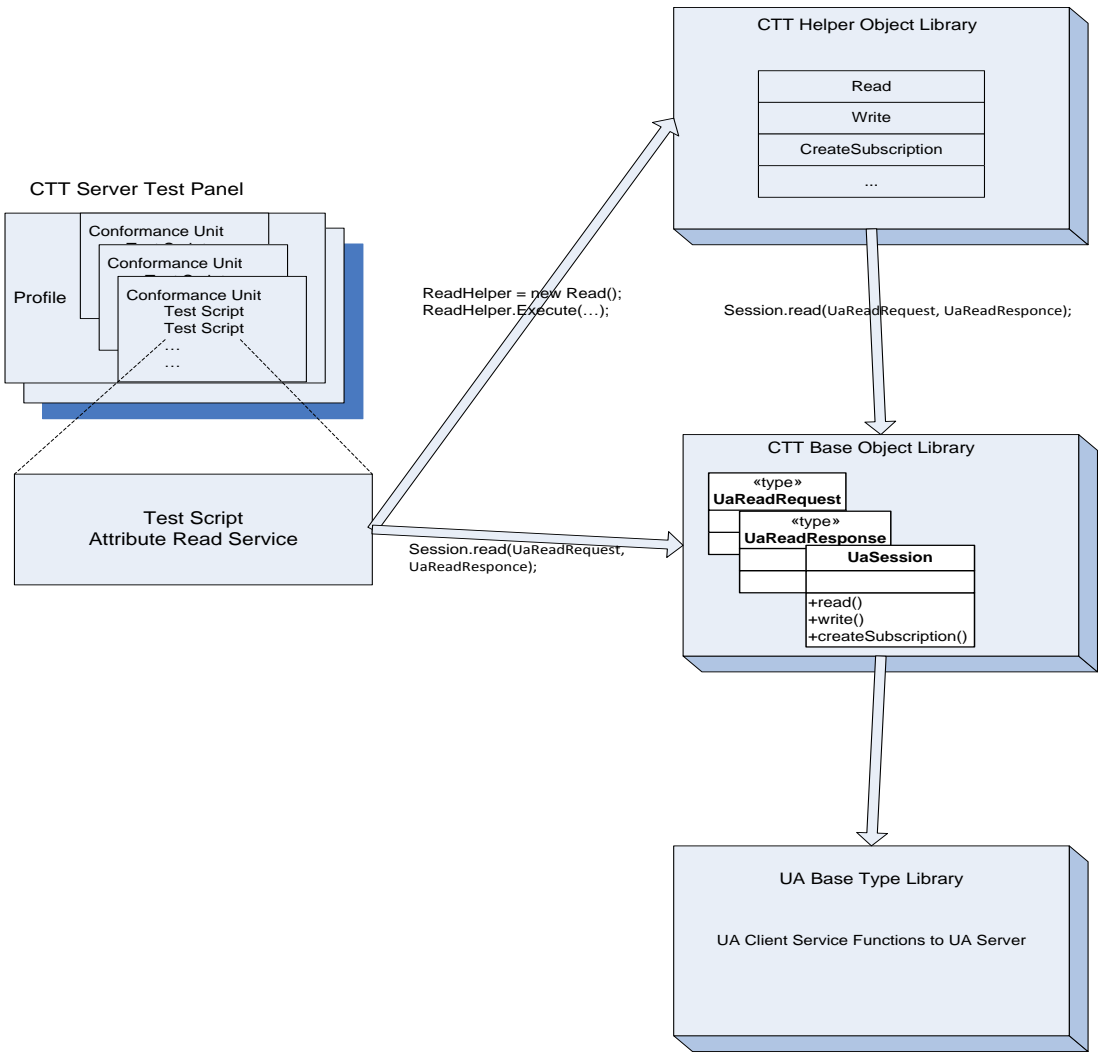
CTT type libraries

There are three kinds of Script libraries implemented in the UA CTT environment. CTT Helper Object Library and CTT Base Object Library are written in Java Script. UA Base Type Library is an embedded library written in C++. Each of them defines object types served for different purposes:

Library	Description	Source code available?
CTT Helper Object Library	<p>Helper objects defined to implement the test cases for Conformance Units and UA Service Sets. It is a high level function library called by test scripts and manages objects defined in CTT Base Object Library. The helper types are also created to performance repeated items such as checking a common structure or handling some error etc.</p> <p>Example: Read – the Execute method of Read Helper object calls the read service function of UA Attribute Service Set in CTT Base Object Library.</p>	Yes
CTT Base Object Library	<p>CTT Java based objects are defined equivalently to Base objects defined in UA Base Type Library. The object combines inputs from CTT Helper object and configuration settings from the test project to call the UA service functions implemented in UA Base Type Library.</p> <p>Example: In Read Helper type, objects of UaReadRequest, UaReadResponce and the method UaSession.read in CTT Base Object Library are declared to implement Read.Execute method.</p>	No
UA Base Type Library	Implemented as UA framework to support UA	No

	Server/Client services. It is an embedded class library.	
--	--	--

The following example of a UA Server attribute read illustrates the relations among mentioned libraries:



To write a test script for a specific Conformance Unit, you can use either CTT Helper Object or CTT Base Object to call UA service functions. CTT Helper Object Library provides the convenient functions wrapping the construction of CTT Base Objects within. CTT Base Object makes the direct call with the familiar OPC UA service function pattern; however, it requires the knowledge of CTT Base Object type.

In this document, the programming notes regarding to use CTT Helper and Base objects are discussed.

Safe Invocation of Test Script

Safely invokes a named method within a Try/Catch/Finally block that ensures all exceptions thrown in test script are caught and the method is finalized gracefully.

safelyInvoke

Description: This method prototypes methods in try, catch and finally blocks.

Include: `"./library/Base/safelyInvoke.js"`

Base Object: global function

Method: *safelyInvoke(methodInTry, methodInCatch, methodInFinally)*

Code Snippet: `safelyInvoke(read581001);`

Initialization/Cleanup

Initialization applies to every test script. It is implemented in initialize.js file. In a typical initialization phrase, the following actions are completed:

- The communication to the UA Server is established. It includes the creation of a channel, a session object and the connection to the server. The channel and session resources can be reused in the sequential test cases in the specific conformance unit.
- The test nodes are populated.
- The Helper objects are created.
- The preliminary tests are conducted.

After the test script is run, a cleanup procedure is required to match the resource allocation in Initialize function. Usually, it is implemented in cleanup.js file and executed automatically in CTT runtime environment. The cleanup includes actions below:

- Delete any global resources created in Initialization.
- Cleanup Helper objects.
- Restore original states in test nodes.
- Disconnect from UA server.
- Cleanup created CTT Base Type objects.

Create Channel

Base Object: `UaChannel`

Method: *UaChannel()*

Code Snippet: `var g_channel = new UaChannel();`

Create Session

Base Object: `UaSession`

Method: *UaSession(UaChannel)*

Code Snippet: `var g_session = new UaSession(g_channel);`

Connect to sever

Description: This function object establishes the communication connection to UA Server and CTT test program. It populates the properties in UaChannel object and connects channel. Then, it creates the session with UA Server and activates the created session.

Include: `"./library/Base/connect.js"`

Base Object: `global function`

Method: *connect(UaChannel, UaSession)*

Code Snippet: `if(!connect(g_channel, g_session))
{addError("Connect failed.");}`

Disconnect from server

Description: This function object cleans up the communication connection to UA Server and CTT test program. It closes the session to the server and disconnects the communication channel.

Include: `"./library/Base/disconnect.js"`

Base Object: `global function`

Method: *disconnect(UaChannel, UaSession)*

Code Snippet: `disconnect(g_channel, g_session);`

NodeId, Item and Setting Management

NodeId Management provides the convenient functions to read the project setting. The MonitoredItem object has methods to load the values from the specified settings. The global function readSetting provides the base function to retrieve the value of any specified project setting.

Read Defined NodeIds from Project Settings

Description: NodeIdSettings provides a group of methods to retrieve nodes configuration from project settings. In a typical method call, the name of method implies the targeted nodes.

Include: `"/library/Base/SettingsUtilities/NodeIds.js"`

Base Object: NodeIdSettings

Method: *NodeIdSettings .ScalarStatic()*
NodeIdSettings .ScalarStatic1OneType()
NodeIdSettings .ScalarStaticAll()
NodeIdSettings .ScalarDynamic()
NodeIdSettings .ScalarDynamic1OneType()
NodeIdSettings .ScalarDynamicAll()
NodeIdSettings .ArraysStatic()
NodeIdSettings .ArraysDADeadband()
NodeIdSettings .DAStaticDataItem()
NodeIdSettings .DAStaticAnalog()
NodeIdSettings .DAStaticTwoStateDiscreteItems()
NodeIdSettings .DAStaticMultiStateDiscreteItems()
NodeIdSettings .Paths()
NodeIdSettings .getNodeWithReferencesInBothDirections()
NodeIdSettings .UnknownNodeIds()
NodeIdSettings .InvalidNodeIds()
NodeIdSettings .NodeClasses()
NodeIdSettings .AdvancedInvalid()
NodeIdSettings .GetScalarStaticNodeIds(maxNumberNeeded)
NodeIdSettings .GetScalarStaticArrayNodeIds(maxNumberNeeded)
NodeIdSettings .GetDAAnalogStaticNodeIds(maxNumberNeeded)
NodeIdSettings .GetAScalarNodeIdSetting(settings, preferredDatatypesIn)
NodeIdSettings .GetAScalarStaticNodeIdSetting(preferredDatatypesIn)
NodeIdSettings .GetScalarDynamicNodeIds()
NodeIdSettings .GetAScalarDynamicNodeIdSetting(preferredDatatypesIn)
NodeIdSettings .GetAStaticAnalogNodeIdSetting()
NodeIdSettings .GetArrayStaticNodeIds()
NodeIdSettings .GetArrayDynamicNodeIds()

Code Snippet: `var settingNames = NodeIdSettings.ScalarStaticAll();`

Read settings by readSetting function

Base Object: Global function

Method: *readSetting(name)*

Code Snippet: `var settingName = NodeIdSettings.ScalarStatic();
var settingValue = readSetting(settingName);`

Attribute Services

Attribute Services provides the methods to engage UA Attribute Read/Write service calls to UA Server. The services can be performed by methods available from Helper objects or UaSession object.

Attribute Read Service by Read Helper

Description: Read values of items by Read Helper object

Include: `"/library/ServiceBased/AttributeServiceSet/Read/read.js"`

Base Object: Read
Method: *Read(session)*
Execute(monitoredItems, timestampsToReturn, maxAge, expectedErrors, expectErrorNotFail)
setMonitoredItemValues(monitoredItems)
ValuesToString(MaxStringSize)
Code Snippet:

```
var ReadHelper = new Read( g_session );
var monitoredItems = MonitoredItem.fromSettings( NodeIdSettings.ScalarStatic() );
if( ReadHelper.Execute( monitoredItems[0], TimestampsToReturn.Server ) )
{ addLog( "Read Results:\n\t" + ReadHelper.ValuesToString() ); }
```

Attribute Read Service by Session object

Description: Read values of items by Base Type object UaSession
Base Object: UaSession
Method: *read(UaReadRequest, UaReadResponse)*
Code Snippet:

```
var readReq = new UaReadRequest();
var readResp = new UaReadResponse();
// g_session is a connected UaSession object
g_session.buildRequestHeader( readReq.RequestHeader );
// Setup NodeId, Attribute, other properties in readReq
// uaStatus is a UaStatusCode object
uaStatus = g_session.read( readReq, readResp );
if( uaStatus.isGood() ){} else {}
```

Attribute Write Service by Write Helper

Description: Write values of items by Write Helper object
Include: `"/library/ServiceBased/AttributeServiceSet/ Write/write.js"`
Base Object: Write
Method: *Write(session)*
Execute(monitoredItems, expectedErrors, checkWriteErr, checkNotSupported, skipVerification)
Code Snippet:

```
// g_session is a connected UaSession object
var WriteHelper = new Write( g_session );
var items = MonitoredItem.fromSettings( NodeIdSettings.ArraysStatic(), 0, Attribute.Value );
// Setup value of Value of item in items
for( var i=0; i<items.length; i++ )
{
    GenerateScalarValue( items[i].Value.Value, NodeIdSettings.guessType( items[i].NodeSetting ), 2
    );
}
//Write values to server
WriteHelper.Execute(items);
```

Attribute Write Service by Session object

Description: Write values of items by Base Type object UaSession
Base Object: UaSession
Method: *write(UaWriteRequest, UaWriteResponse)*
Code Snippet:

```
var writeReq = new UaWriteRequest();
var writeResp = new UaWriteResponse();
// g_session is a connected UaSession object
g_session.buildRequestHeader(writeReq.RequestHeader );
// Setup NodeId, Attribute, Value, Index, other properties in writeReq
// uaStatus is a UaStatusCode object
uaStatus = g_session.write(writeReq, writeResp );
if( uaStatus.isGood() ){} else {}
```

Discovery Service Set

Discovery Services provides the methods to get endpoints, search servers and register a UA Server process.

Discovery Service – Discovery object

Base Object: UaDiscovery

Method: *UaDiscovery(UaChannel)*

Code Snippet: `var g_discovery = new UaDiscovery(g_channel);`

Discovery Service – Get Endpoints by Helper object

Include: `"./library/ServiceBased/DiscoveryServiceSet/GetEndpoints/getEndpoint.js"`

Base Object: GetEndpoints

Method: *GetEndpoints(session)*

Execute(endpointUrl, locales, serverUris, expectedErrors)

Code Snippet: `var endpointUrl = readSetting("/Server Test/Discovery URL");
var localeIds = new UaStrings();
var serverUris = readSetting("/Server Test/Server URL");
var GetEndpointsHelper = new GetEndpoints (g_session);
if(!GetEndpointsHelper.Execute(endpointUrl, localeIds, serverUris))
{ addError("GetEndPoints failed.");};`

Discovery Service – Get Endpoints

Base Object: UaDiscovery

Method: *getEndpoints(UaGetEndpointsRequest, UaGetEndpointsResponse)*

Code Snippet: `var getEndpointsRequest = new UaGetEndpointsRequest();
var getEndpointsResponse = new UaGetEndpointsResponse();
var uaStatus = g_discovery.getEndpoints(getEndpointsRequest, getEndpointsResponse);
if(uaStatus.isBad()) {} else {}`

Discovery Service – Find Servers by Helper object

Include: `"./library/ServiceBased/DiscoveryServiceSet/FindServers/findServers.js"`

Base Object: FindServers

Method: *FindServers(session)*

Execute(endpointUrl, locales, serverUris, expectedErrors)

Code Snippet: `var endpointUrl = readSetting("/Server Test/Discovery URL");
var localeIds = new UaStrings();
var serverUris = readSetting("/Server Test/Server URL");
var FindServersHelper = new FindServers (g_session);
if(!FindServersHelper.Execute(endpointUrl, localeIds, serverUris))
{ addError("FindServers failed.");};`

Discovery Service – Find Servers

Base Object: UaDiscovery

Method: *findServers(UaFindServersRequest, UaFindServersResponse)*

Code Snippet: `var findServersRequest = CreateDefaultFindServersRequest();
var findServersResponse = new UaFindServersResponse();
var uaStatus = g_discovery.findServers(findServersRequest, findServersResponse);
if(uaStatus.isBad()) {} else {}`

Discovery Service – RegisterServer by Helper object

Include: `"./library/ServiceBased/DiscoveryServiceSet/RegisterServer/registerServer.js"`

Base Object: RegisterServer

Method: *RegisterServer(session)*
Execute(discoveryUrl, gatewayServerUri, isOnline, productUri, semaphorePath, serverName, appType, serverUri, expectedErrors)

Code Snippet:

```
var discoveryUrl = readSetting("/Server Test/Discovery URL");
var RegisterServerHelper = new RegisterServer( g_session );
if(!RegisterServerHelper.Execute(discoveryUrl)
{ addError("RegisterServer failed.");});
```

Discovery Service – RegisterServer

Base Object: UaDiscovery

Method: *registerServer UaRegisterServerRequest, UaRegisterServerResponse)*

Code Snippet:

```
var registerServerRequest = new UaRegisterServerRequest();
var registerServerResponse = new UaRegisterServerResponse();
//Setup registerServerRequest properties
g_discovery.registerServer( registerServerRequest,registerServerResponse );
if( uaStatus.isBad() ) {} else {}
```

View Service Set

View Service Set implements the CTT functionality of UA Browse Services to UA Server.

View Service – Browse by BrowseHelp object

Include: *"/library/ServiceBased/ViewServiceSet/Browse/Browse.js"*

Base Object: Browse

Method: *Browse(session)*
Execute(nodesToBrowse, maxRefsToReturn, view, expectedErrors, expectErrorNotFail)
ResultsToString()

Code Snippet:

```
var BrowseHelper = new Browse(g_session);
var m1 = MonitoredItem.fromSetting( "/Server Test/NodeIds/References/Has 3 Forward References 1" );
// Set browse filter properties: both direction, w/subtypes, all node classes, all results
m1.SetBrowse( BrowseDirection.Both, true, 0xff, null, BrowseResultMask.All);
// Expecting/assuming that the node specified in the settings has 1+ references
if( BrowseHelper.Execute( m1 ) ){ print( BrowseHelper.ResultsToString() );}
```

View Service – Browse by Session object

Base Object: UaSession

Method: *browse (UaBrowseRequest, UaBrowseResponse)*

Code Snippet:

```
var request = new UaBrowseRequest();
var response = new UaBrowseResponse();
// set up UaBrowseRequest object
var uaStatus = g_session.browse(request, response);
if( uaStatus.isBad() ) {} else {}
```

View Service – BrowseNext by BrowseNext object

Include: *"/library/ServiceBased/ViewServiceSet/BrowseNext/BrowseNext.js"*

Base Object: BrowseNext

Method: *Browsenext(session)*
Execute (continuationPoints, releaseCPs, expectedErrors, expectErrorNotFail)
ResultsToString()

Code Snippet:

```
var BrowseNextHelper = new BrowseNext(g_session);
var contPoints = MonitoredItem.fromSetting( "/Server Test/NodeIds/References/Has 3 Forward
References 1" );
// we are expecting/assuming that the node specified in the settings has 1+ references
```

```

if( BrowseNextHelper.Execute(contPoints) )
    { print( BrowseNextHelper.ResultsToString() );

```

View Service – BrowseNext by Session object

Base Object: UaSession
 Method: *browseNext(UaBrowseNextRequest, UaBrowseNextResponse)*
 Code Snippet:

```

var request = new UaBrowseNextRequest();
var response = new UaBrowseNextResponse();
// Setup ContinuationPoints property in request
var uaStatus = g_session. browseNext(request, response);
if( uaStatus.isBad() ) {} else {}

```

View Service – RegisterNodes by Session object

Base Object: UaSession
 Method: *registerNodes(UaRegisterNodesRequest, UaRegisterNodesResponse)*
 Code Snippet:

```

var request = new UaRegisterNodesRequest(g_session);
var response = new UaRegisterNodesResponse ();
// set up UaRegisterNodesRequest object
var uaStatus = g_session. registerNodes (request, response);
if( uaStatus.isBad() ) {} else {}

```

View Service – UnRegisterNodes by Session object

Base Object: UaSession
 Method: *unregisterNodes(UaUnregisterNodesRequest, UaUnregisterNodesResponse)*
 Code Snippet:

```

var request = new UaUnregisterNodesRequest(g_session);
var response = new UaUnregisterNodesResponse ();
// set up UaUnregisterNodesRequest object
var uaStatus = g_session. unregisterNodes (request, response);
if( uaStatus.isBad() ) {} else {}

```

View Service – TranslateBrowsePathsToNodeIds by Helper object

Include: `"/library/ServiceBased/ViewServiceSet/TranslateBrowsePathsToNodeIdsHelper/translateBrowsePathsToNodeIdsHelper.js"`
 Base Object: TranslateBrowsePathsToNodeIdsHelper
 Method: *TranslateBrowsePathsToNodeIdsHelper()*
executeAndValidate(session)
 Code Snippet:

```

var TBPHelper = new TranslateBrowsePathsToNodeIdsHelper();
TBPHelper. executeAndValidate(g_session);

```

View Service – TranslateBrowsePathsToNodeIds by Session object

Base Object: UaSession
 Method: *translateBrowsePathsToNodeIds (UaTranslateBrowsePathsToNodeIdsRequest, UaTranslateBrowsePathsToNodeIdsResponse)*
 Code Snippet:

```

var request = new UaTranslateBrowsePathsToNodeIdsRequest (g_session);
var response = new UaTranslateBrowsePathsToNodeIdsResponse ();
// set up UaTranslateBrowsePathsToNodeIdsRequest object
var uaStatus = g_session. translateBrowsePathsToNodeIds (request, response);
if( uaStatus.isBad() ) {} else {}

```

MonitoredItem Service Set

MonitoredItem Services provide CTT enabled UA MonitoredItem Client Services to a UA Server.

MonitoredItem Service – MonitoredItem object

Description: Client side object in CTT to set up a monitored item in UA Server in MonitoredItem Services.

Include: `"/library/Base/Objects/MonitoredItem.js"`

Base Object: `MonitoredItem`

Method: *MonitoredItem (nodeId, clientHandle, attributeId, indexRange, monitorMode, discardOldest, filter, queue, interval, timestampsToReturn, nodeSetting)*
SetBrowse (direction, subtypes, classMask, referenceTypeId, resultsMask)
MonitoredItem. fromNodeIds (Nodes, attributeId, indexRange, monitorMode, discardOldest, filter, queue, interval, timestampsToReturn)
MonitoredItem. fromSetting(settingName, clientHandle, attributeId, indexRange, monitorMode, discardOldest, filter, queue, interval, timestampsToReturn)

Code Snippet: `var items = MonitoredItem.fromSettings(NodeIdSettings.ScalarStatic(),0);`

MonitoredItem Service – CreateMonitoredItems by Helper object

Include: `"/library/ServiceBased/MonitoredItemServiceSet/CreateMonitoredItems/ createMonitoredItems.js"`

Base Object: `global function`

Method: *createMonitoredItems(MonitoredItems, TimestampsToReturn, Subscription, Session, ExpectedResults, ExpectErrorNotFail)*

Code Snippet: `//Collect NodeIds for monitored items
var items = MonitoredItem.fromSettings (NodeIdSettings.ScalarStatic(),0);
//Create a subscription
var MonitorBasicSubscription = new Subscription();
if(createSubscription(MonitorBasicSubscription, g_session))
{
 //Create MonitoredItems
 createMonitoredItems(items,TimestampsToReturn.Server,
 MonitorBasicSubscription,g_session);
}`

MonitoredItem Service – CreateMonitoredItems by Session object

Base Object: `UaSession`

Method: *createMonitoredItems(UaCreateMonitoredItemsRequest, UaCreateMonitoredItemsResponse)*

Code Snippet: `var request = new UaCreateMonitoredItemsRequest();
var response = new UaCreateMonitoredItemsResponse();
// set up properties in UaCreateMonitoredItemsRequest object
var uaStatus = g_session. createMonitoredItems (request, response);
if(uaStatus.isBad()) {} else {}`

MonitoredItem Service – DeleteMonitoredItems by Helper object

Description: Delete all monitoredItems from the subscription

Include: `"/library/ServiceBased/MonitoredItemServiceSet/DeleteMonitoredItems/ deleteMonitoredItems.js"`

Base Object: `global function`

Method: *deleteMonitoredItems(MonitoredItemIds, Subscription, Session, ExpectedErrors, ExpectErrorNotFail)*

Code Snippet: `deleteMonitoredItems(items, MonitorBasicSubscription, g_session);`

MonitoredItem Service – DeleteMonitoredItems by Session object

Description: Delete all monitoredItems from the subscription

Base Object: `UaSession`

Method: *createMonitoredItems(UaDeleteMonitoredItemsRequest, UaDeleteMonitoredItemsResponse)*

Code Snippet: `var request = new UaDeleteMonitoredItemsRequest();
var response = new UaDeleteMonitoredItemsResponse();
// set up properties in UaDeleteMonitoredItemsRequest object
var uaStatus = g_session. deleteMonitoredItems (request, response);`


```
if( uaStatus.isBad() ) {} else {}
```

MonitoredItem Service – ModifyMonitoredItems by Helper object

Description: Modify monitoredItems from the subscription

Base Object: ModifyMonitoredItemsHelper

Method: *ModifyMonitoredItemsHelper(sessionObject)*

Execute(itemsToModify, timestampsToReturn, subscriptionObject, expectedResults, errorNotFail)

```
Code Snippet: var ModifyMonitoredItemsHelper = new ModifyMonitoredItemsHelper( g_session );
               var items = MonitoredItem.fromSetting( "SomeNodeSettingName", 0 );
               var MonitorBasicSubscription = new Subscription( );
               if( createSubscription( MonitorBasicSubscription, g_session ) )
               {
                   if( !createMonitoredItems( items, TimestampsToReturn.Both, MonitorBasicSubscription,
                   g_session ) )
                   {
                       if( !ModifyMonitoredItemsHelper.Execute( item, TimestampsToReturn.Both,
                       MonitorBasicSubscription ) ) {}
                   }
               }
               }
```

MonitoredItem Service – ModifyMonitoredItems by Session object

Description: Modify monitoredItems from the subscription

Base Object: UaSession

Method: *modifyMonitoredItems(UaModifyMonitoredItemsRequest, UaModifyMonitoredItemsResponse())*

```
Code Snippet: var request = new UaModifyMonitoredItemsRequest();
               var response = new UaModifyMonitoredItemsResponse();
               // set up properties in UaModifyMonitoredItemsRequest object
               var uaStatus = g_session. modifyMonitoredItems (request, response );
               if( uaStatus.isBad() ) {} else {}
```

MonitoredItem Service – SetMonitoringMode by Helper object

Description: Set monitoring mode

Base Object: SetMonitoringMode

Method: *SetMonitoringMode (sessionObject)*

Execute (monitoringMode, monitoredItems, subscription, expectedErrors, expectErrorNotFail)

```
Code Snippet: var items = MonitoredItem.fromSetting( "SomeNodeSettingName", 0 );
               var subscription = new Subscription();
               if(createSubscription(subscription1,g_session))
               {
                   if( createMonitoredItems( items, TimestampsToReturn.Both, subscription, g_session ) )
                   {
                       var setMonitoringModeHelper = new SetMonitoringMode(g_session );
                       if(setMonitoringModeHelper.Execute(MonitoringMode.Disabled,items,
                       subscription)){ }
                   }
               }
               }
```

MonitoredItem Service – SetMonitoringMode by Session object

Description: Set monitoring mode

Base Object: UaSession

Method: *setMonitoringMode (UaSetMonitoringModeRequest, UaSetMonitoringModeResponse)*

```
Code Snippet: var request = new UaSetMonitoringModeRequest ();
               var response = new UaSetMonitoringModeResponse();
```

```
// set up properties in UaSetMonitoringModeRequest object
var uaStatus = g_session.setMonitoringMode (request, response );
if( uaStatus.isBad() ) {} else {}
```

MonitoredItem Service – SetTriggering by Helper object

Description: Set a triggering link

Base Object: SetTriggering

Method: *SetTriggering (session)*

Execute (subscription, triggeringItem, linksToAdd, linksToDelete, expectErrorNotFail, expectedErrorsAdd, expectedErrorsDelete)

```
Code Snippet: var subscription = new Subscription();
if( !createSubscription( subscription, g_session ) ) return;
var triggeringItem = MonitoredItem.fromSetting( "setting1" );
var addLinkedItem = MonitoredItem.fromSetting( "setting2" );
if( createMonitoredItems( [triggeringItem, addLinkedItem], TimestampsToReturn.Both, subscription,
g_session ) )
{
    var setTriggeringHelper = new SetTriggering( g_session );
    if( setTriggeringHelper.Execute( MonitorTriggeringSubscription, triggeringItem,
[addLinkedItem]) ) { }
```

MonitoredItem Service – SetTriggering by Session object

Description: Set a triggering link

Base Object: UaSession

Method: *setTriggering(UaSetTriggeringRequest, UaSetTriggeringResponse)*

```
Code Snippet: var request = new UaSetTriggeringRequest();
var response = new UaSetTriggeringResponse();
// set up properties in UaSetTriggeringRequest object
var uaStatus = g_session.setTriggering (request, response );
if( uaStatus.isBad() ) {} else {}
```

Subscription Service Set

Subscription Services in CTT provides the Client subscription service of monitored items to a UA Server.

Subscription objects

Include: `"/library/Base/Objects/subscription.js"`

Base Object: Subscription

Method: *Subscription (publishingInterval, publishingEnabled, requestedLifetimeCount, requestedMaxKeepAliveCount, maxNotificationsPerPublish, priority)*
SetParameters (publishingInterval, publishingEnabled, requestedLifetimeCount, requestedMaxKeepAliveCount, maxNotificationsPerPublish, priority)

```
Code Snippet: var subscription = new Subscription();
```

Subscription Service – CreateSubscription by Helper object

Description: Create a Subscription

Include: `"/library/ServiceBased/SubscriptionServiceSet/CreateSubscription/ createSubscription.js"`

Base Object: global function

Method: *createSubscription (Subscription, Session, expectedErrors)*

```
Code Snippet: var subscription = new Subscription();
if( createSubscription( subscription, g_session ) ) {}
```

Subscription Service – CreateSubscription by Session object

Description: Create a Subscription

Base Object: UaSession

Method: *createSubscription (UaCreateSubscriptionRequest, UaCreateSubscriptionResponse)*

```
Code Snippet: var request = new UaCreateSubscriptionRequest ();
               var response = new UaCreateSubscriptionResponse();
               // set up properties in UaCreateSubscriptionRequest object
               var uaStatus = g_session.createSubscription (request, response );
               if( uaStatus.isBad() ) {} else {}
```

Subscription Service – DeleteSubscription by Helper object

Description: Delete a Subscription

Include: `"./library/ServiceBased/SubscriptionServiceSet/DeleteSubscription/ deleteSubscription.js"`

Base Object: global function

Method: *deleteSubscription (Subscription, Session, expectedErrors)*

```
Code Snippet: var subscription = new Subscription();
               if( deleteSubscription( subscription, g_session ) ) {}
```

Subscription Service – DeleteSubscription by Session object

Description: Delete a Subscription

Base Object: UaSession

Method: *deleteSubscriptions(UaDeleteSubscriptionsRequest, UaCreateSubscriptionResponse)*

```
Code Snippet: var request = new UaDeleteSubscriptionsRequest();
               var response = new UaDeleteSubscriptionsResponse();
               // set up properties in UaDeleteSubscriptionsRequestobject
               var uaStatus = g_session. deleteSubscriptions (request, response );
               if( uaStatus.isBad() ) {} else {}
```

Subscription Service – ModifySubscription by Helper object

Description: Modify a Subscription

Include: `"./library/ServiceBased/SubscriptionServiceSet/ModifySubscription/ modifySubscription.js"`

Base Object: ModifySubscription

Method: *ModifySubscription(session)*
Execute (subscription, expectedErrors)

```
Code Snippet: var subscription = new Subscription();
               if( createSubscription( subscription, g_session ) )
               {
                   var modifySubscriptionHelper = new ModifySubscription( g_session );
                   subscription.SetParameters( 2000, true, 30, 10, 0, 0 );
                   modifySubscriptionHelper.Execute( subscription );
               } {} else {}
```

Subscription Service – ModifySubscription by Session object

Description: Modify a Subscription

Base Object: UaSession

Method: *modifySubscription (UaModifySubscriptionRequest, UaModifySubscriptionResponse)*

```
Code Snippet: var request = new UaModifySubscriptionRequest ();
               var response = new UaModifySubscriptionResponse ();
               // set up properties in UaModifySubscriptionRequest object
               var uaStatus = g_session. modifySubscription (request, response );
               if( uaStatus.isBad() ) {} else {}
```

Subscription Service – SetPublishingMode by Helper object

Description: Enable/Disable publish mode

Include: “./library/ServiceBased/SubscriptionServiceSet/SetPublishingMode/setPublishingMode.js”

Base Object: SetPublishingMode

Method: *SetPublishingMode (session)*
Execute (subscriptions, publishingEnabled, expectedErrors, errorExpected)

```
Code Snippet: var subscription = new Subscription();
               if( createSubscription( subscription, g_session ) )
               {
                   var setPublishing = new SetPublishingMode(g_session);
                   if( setPublishing.Execute(subscriptions, true))
                   {} else {}
               }
```

Subscription Service – SetPublishingMode by Session object

Description: Enable/Disable publish mode

Base Object: UaSession

Method: *setPublishingMode (UaSetPublishingModeRequest, UaSetPublishingModeResponse)*

```
Code Snippet: var request = new UaSetPublishingModeRequest ();
               var response = new UaSetPublishingModeResponse ();
               // set up properties in UaSetPublishingModeRequest object
               var uaStatus = g_session. setPublishingMode (request, response );
               if( uaStatus.isBad() ) {} else {}
```

Subscription Service – Publish by Helper object

Description: Request publish service

Include: “./library/ServiceBased/SubscriptionServiceSet/Publish/publish.js”

Base Object: Publish

Method: *Publish (session, timeoutHint)*
Execute (noAcks, expectedErrors, expectErrorNotFail)

```
Code Snippet: var subscription = new Subscription();
               if( createSubscription( subscription, g_session ) )
               {
                   var publishService = new Publish( g_session );
                   if(publishService.Execute())
                   {} else {}
               }
```

Subscription Service – Publish by Session object

Description: Request publish service

Base Object: UaSession

Method: *publish(UaPublishRequest, UaPublishResponse)*

```
Code Snippet: var request = new UaPublishRequest ();
               var response = new UaPublishResponse ();
               // set up properties in UaPublishRequest object
               var uaStatus = g_session. publish (request, response );
               if( uaStatus.isBad() ) {} else {}
```

Subscription Service – Transfer Subscription by Helper object

Description: Transfer subscription service

Include: “./library/ServiceBased/SubscriptionServiceSet/ TransferSubscriptions / transferSubscriptions.js”

Base Object: `transferSubscriptionsHelper`
Method: `transferSubscriptionsHelper(sourceSession, destinationSession, subscriptions)`
`Execute ()`
Code Snippet:

```
// g_session1, g_session2 are active sessions
var subscription = new Subscription();
if( createSubscription( subscription, g_session1 ) )
{
    var transferHelper = new transferSubscriptionsHelper(g_session1, g_session2, [subscription]);
    transferHelper.Request.SendInitialValues = true;
    var uaStatus = transferHelper.Execute();
    if( uaStatus.isBad() ) {} else {}
}
```

Subscription Service – Transfer Subscription by Session object

Description: `Transfer subscription service`
Base Object: `UaSession`
Method: `transferSubscriptions(UaTransferSubscriptionsRequest, UaTransferSubscriptionsResponse)`
Code Snippet:

```
var request = new UaTransferSubscriptionsRequest ();
var response = new UaTransferSubscriptionsResponse ();
// set up properties in UaTransferSubscriptionsRequest object
var uaStatus = g_session. transferSubscriptions (request, response );
if( uaStatus.isBad() ) {} else {}
```

Summary

Helper Object	Method	Conformance Unit	CTT Base Object Method
global function	safelyInvoke	Apply to all	Client local function
Browse	Execute	View Service	UaSession.browse
BrowseNext	Execute	View Service	UaSession.browseNext
FindServers	Execute	Discovery Service	UaDiscovery.findServers
GetEndpoints	Execute	Discovery Service	UaDiscovery.getEndpoints
global function	connect	Apply to all	UaChannel.connect
global function	disconnect	Apply to all	UaChannel.disconnect
global function	readSetting	Apply to all	Client local function
global function	createMonitoredItems	MonitoredItem Service	UaSession.createMonitoredItems
global function	DeleteMonitoredItems	MonitoredItem Service	UaSession.deleteMonitoredItems
global function	createSubscription	Subscription Service	UaSession.createSubscription
global function	deleteSubscription	Subscription Service	UaSession.deleteSubscription
ModifyMonitoredItemsHelper	Execute	MonitoredItem Service	UaSession.modifyMonitoredItems
ModifySubscription	Execute	Subscription Service	UaSession.modifySubscription
MonitoredItem	MonitoredItem.fromSettings	Attribute Services	Client local function
MonitoredItem	SetBrowse	Attribute Services	Client local function
MonitoredItem	MonitoredItem.fromNodeIds	MonitoredItem Service	Client local function
NodeIdsSettings	NodeIdsSettings.*	Apply to all	Client local function
Publish	Execute	Subscription Service	UaSession.publish
Read	Execute	Attribute Read Service	UaSession.read
RegisterServer	Execute	Discovery Service	UaDiscovery.registerServer
SetMonitoringMode	Execute	MonitoredItem Service	UaSession.setMonitoringMode
SetPublishingMode	Execute	Subscription Service	UaSession.setPublishingMode
SetTriggering	Execute	MonitoredItem Service	UaSession.setTriggering
transferSubscriptionsHelper	Execute	Subscription Service	UaSession.transferSubscriptions
TranslateBrowsePathsToNodeIdsHelper	executeAndValidate	View Service	UaSession.translateBrowsePathsToNodeIds
UaSession	registerNodes	View Service	UaSession.registerNodes
UaSession	unregisterNodes	View Service	UaSession.unregisterNodes
Write	Execute	Attribute Write Service	UaSession.write

Conclusion

Having an automate test tool to validate UA Server functionality and to ensure the quality of software is the goal for using the UA Test Application and CTT. Both provide the foundation and framework for developers to make enhancements. A successful run through both test tools can confirm some degree of quality in the custom UA Server. The UA Test Application validates the essential UA Services. The CTT provides the official test procedures to qualify the target UA Server.

However, even with OPC provided Sample Server, the CTT had errors due to missing configuration and unsupported features. There are also a lot of “ToDo” – un-implemented test procedures in CTT. The UA Test Application requires the developer to work on source code in VS 2008 application project, make calls to UA Client Service operations directly in C#. For a seasoned UA developer, it may be trivial; however, there is no leverage from the existing implementation. To add a new test case in CTT, the developer writes the test script in JavaScript to interface the service helper and base libraries provided by OPC Foundation. It is suitable for more audiences including non-C# developer. Once the contents of the CTT library is understood, the test generator has the ability to write the test procedure(s) without worrying about the UA Service plumbing.