OPC Testbench

Introduction

The OPC testbench has been developed within EN-ICE to provide an automated means of testing OPC Servers. The idea is to build and maintain a suite of tests that will be routinely applied to any (EN-ICE supported) OPC server release to provide a baseline assurance that an OPC server works.

Automated testing is a fairly well established software development practice, the headline benefits being that tests are repeatable and cost less to run because machines do it, not people. Note: A little 'by-hand' testing is probably inevitable, some esoteric tests may be just too hard to script and it may be more efficient for a person to execute these. The golden rule is this: Any test that can be automated, should.

The Testbench Components

Testing an OPC server with the test bench requires 2 key components:

1. **The test script.** This is a script written in the OPC testing **DSL**. A script consists of commands which are to be executed against the target OPC server and unit-test style assertions which must be met for the test to be considered a pass. Scripts likely contain many commands and assertions - the general pattern is 'perform an action' then 'assert the results of the action' often this pattern will be inside a loop. Generally each script is customized to a specific OPC namespace: So scripts are vendor specific. A test script designed to test some aspect of the CAEN OPC server will not work against the Wiener OPC server for example, at least not without some modification.

2. **The test script runner.** This is the engine which interprets the script, executes the commands and monitors the assertions to provide pass/fail results. The engine consists of 3 layers:
   a. the script runner, that knows how to drive an OPC client and make assertions
   b. the gui, provides script debug output and reports assertion results: green = pass, red = fail.
   c. the OPC client which drives the target OPC server.

Where do these components come from? The test script runner is an open source project hosted [here](https://github.com) on github. To build it requires maven and MS visual studio and to run it requires only java. Note that the app only builds and runs on windows since the entire concept is based around OPC-DA, which is windows only. The project README.txt contains all the details required for building the project. For test scripts: a suite of test scripts is maintained by EN-ICE but users can write their own, scripts are written in Groovy with extensions as described in this OPC testing DSL documentation [TODO - DSL documentation].

The Grand Plan

It is good to have a grand plan. The grand plan is to express concrete user requirements to OPC server vendors in test scripts, plus words, text and so on but test scripts are key. OPC server vendors install the test script runner on their local systems and modifications or issues with OPC servers will be described (words, text, mime, whatever) and a test script supplied. The idea is that acceptance testing of any changes can be performed at the vendor end before even arriving at CERN - until the supplied tests pass we do not want to see it. Once the supplied test scripts pass then EN-ICE will probably re-run the acceptance test scripts and no doubt additional testing before recommending a version to CERN's user community but the key point is that vendors can verify their changes are acceptable before the new version even leaves the building.

If the OPC-DA testbench is successful then the same idea can be applied to OPC-UA. This will not be a straightforward port however, OPC-UA has a richer 'vocabulary' in terms of how clients can interact with the server and so the DSL for OPC-UA test scripts will likely be larger than the OPC-DA testing DSL in order to allow scripts to test all angles of OPC-UA server implementations.