Testing Overview

The DGP architecture deliberately consolidates all possible functionality in a system into its web service API's. The centralization of functionality greatly improves reusability, simplifies maintenance and deployments, and increases the security of the system.

In short, almost all of the functionality in a DGP system is implemented as web service API methods, all of which can be tested using the built-in API Tester test harness. By focusing on the automated testing of all of these API's, achieving close to 100% code coverage in the total of all the test files in a system becomes feasible.

Compared to standard method calls, the message-based API's allow for the two-way communication of a lot of additional information between the client apps and the web services, and this capability is used to add simple unit test functionality to every API method in the system. This allows the API Tester to call (virtually) every API method in a system end-to-end to verify its security, correct functionality and measure its end-to-end performance – every time a new version of the code is deployed. The test results of any test run can be saved for analysis over time. The beta version of the Lattice API Tester test harness runs a full regression of 218 API method calls (both positive and negative tests) in under 3 seconds.

End-to-End Unit Testing

One of the best capabilities of message-based API's is the two-way communication of data from the client app to the server in the request, and from the server to the client app in the response. This is possible due to the ability of the API request and API response messages to contain a great deal of additional data besides the usual input parameters and method return value. This capability is what makes the API Tester test harness application and the end-to-end unit testing possible. Every API mapper method performs an evaluation of the result returned by the library method it calls. It then maps the result to one of the ResultCode values (OK, Empty, Error, and Exception). The result codes, info and server performance data are returned as part of every API response message.

The combination of both primary path and alternate path testing in the test harness test files offers a good chance to achieve 100% code coverage for every API method tested. The consolidation of all possible functionality of a system into the web service API's allows for 100% code coverage for all of the main functionality in a system. Also, the immutable append-only conventions used for

© 2021 Alan Rahn All Rights Reserved

API methods means that once the test files are working properly, neither the API methods nor the API test files should need any maintenance going forward. This is a key factor in enabling small teams of developers and testers to easily maintain large systems very inexpensively.

From an automation perspective, Continuous Integration/Continuous Deployment/Continuous Testing (CI/CD/CT) processes require some form of automated testing in order to ever be feasible in the first place. This is why testing tools specifically designed for this type of message-based API have been designed and built into DGP systems.

In addition to testing virtually all of the functionality in a system, the API Tester test harness tests the end-to-end performance of every API method as part of each test run, and also allows the test results to be saved to the SysMetrics database of the location being tested. The API Tester test files can be run in dev, test, and QA environments, and can also be run in production environments as long as the test data can be isolated from the rest of the production data. The test files "clean up" after themselves by deleting the test data they create, but those are "soft" deletes and all the test data created are actually still present.

API Tester Test Harness

API Request Message

```
<ReqMsg>
  <UserName />          - DGP system account name
  <ReqID />             - unique ID created by the client app for each request message, and echoed back in the response
  <ReqToken />          - HMAC hash of the Time value using the account password as the secret key
  <Time />              - UTC Unix time of the request for the TTL check and the HMAC hash authentication to the server
  <MList>               - a collection of one or more API methods to be called
    <Meth>
      <MName />         - the name of the API method being called
      <PList>           - a collection of zero or more input parameters for the API method
        <Prm>           - name/value pairs for each input parameter
```


© 2021 Alan Rahn All Rights Reserved

```
        <Name />                    - the name of the input parameter for the API method
        <Val><![CDATA[ … ]]></Val>  - each input parameter value is encapsulated within a CDATA block
      </Prm>
    </PList>
  </Meth>
 </MList>
</ReqMsg>
```

API Response Message

```
<RespMsg>
  <UserName />          - DGP system account name
  <ReqID />             - unique ID created by the client app for each request message, and echoed back in the response
  <Time />              - UTC Unix time of the response
  <Auth />              - state of the request message authentication (OK, NoMatch, Expired, Disabled, Error, Exception)
  <Info />              - optional information regarding Auth states other than OK
  <SvrMS />             - the time spent on the server executing all of the API method calls in the request message batch
  <MethCount />         - the number of methods called in the request message batch
  <RList>               - a variable collection of one or more API method results
    <Result>
      <RName />         - the name of the method result, used by the client to match results to method calls
      <RCode />         - code indicating the state of the method result (OK, Empty, Error, Exception)
      <DType />         - the data type of the result value (Int, Num, Text, DateTime, XML, JSON, DataTable)
      <RVal><![CDATA[ … ]]></Val>   - each return value is encapsulated within a CDATA block
    </Result>
  </RList>
</RespMsg>
```


© 2021 Alan Rahn *All Rights Reserved*

The API Tester test harness runs DGP test files. DGP test files, in turn, are basically API request message XML fragments that are converted into simple templates, which are used to create correctly formatted API request messages when run by the test harness. Test files are a collection of one or more test messages. The core of each test message is an API request message, decorated with additional information needed to define the expected results of the method call, save return values for reuse, etc.

The structure is a bit different from a DGP API request message because each test is run one at a time, sequentially (no batching like in a normal API request message). This is done to allow the results from one test message to be used as the input for subsequent tests in the test file. Using this capability allows test files to form a series of sequential steps, calling different API methods, similar to what a user would do when using an application, etc. The "CRUD" test files use this capability to "clean up" after themselves by deleting the test data that they create as the final steps in their process.

The tests themselves can be thought of as "end-to-end unit tests" thanks to some features built into the DGP web service API's. Specifically, the last step in the mapper methods of the web services analyze the method results returned by the internal method called and assign one of the resultcode values (OK, Empty, Error, Exception) in the API response message. The resultcode is somewhat similar to a unit test assertion, and that is what really makes the API Tester test harness application itself possible. The two-way communication capabilities of the message-based API's are also a necessity, plus the tolerant readers of the API messages in both the client apps and the web services.

DGP Test File

```
<TestBatch>                                          - a collection of one or more test messages
  <TName>APIRole New Negative</TName>
  <TDescrip>Negative tests of the APIRole New method</TDescrip>
  <TGID>{{TGID}}</TGID>
  <TMsg>                                             - each test message is run sequentially by the test harness
    <TMUserName>{{TMUserName}}</TMUserName>          - the double curly braces indicate a template placeholder value
    <TMPassword>{{TMPassword}}</TMPassword>          - placeholder values are replaced with actual values in each test run
    <TMName>APIRole.New.base</TMName>
    <TMDescrip>empty input parameters</TMDescrip>
```

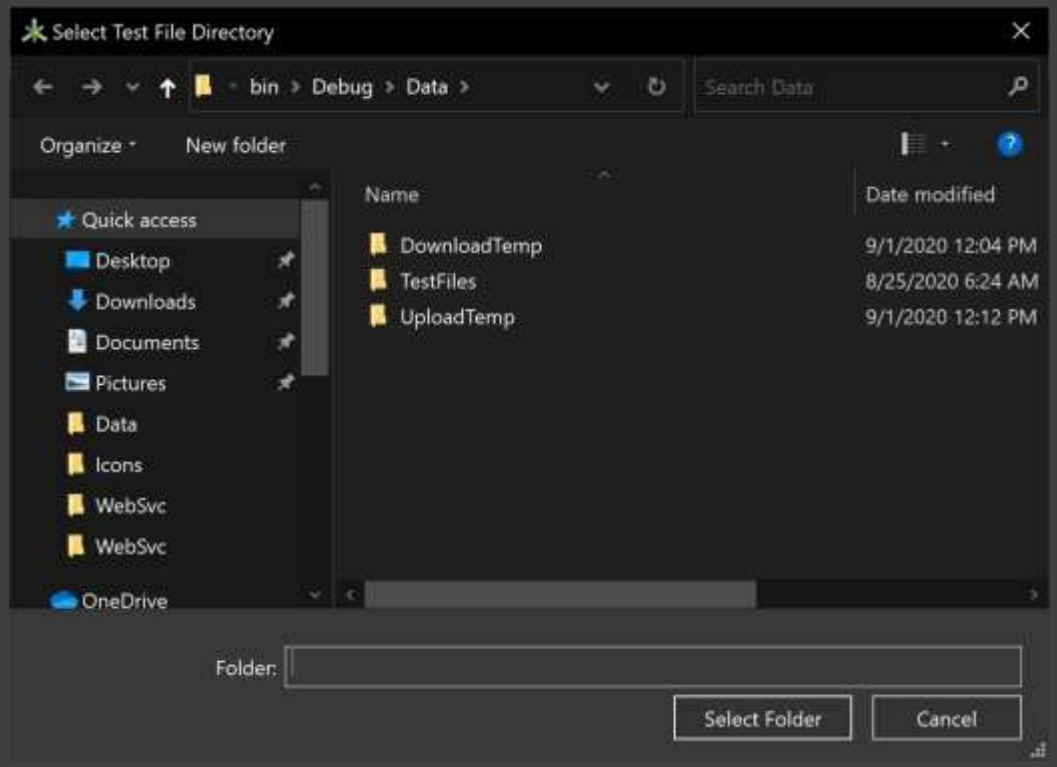

© 2021 Alan Rahn *All Rights Reserved*

```
<TMExpAuthCode>OK</TMExpAuthCode>          - the expected authentication code for the user account
<Meth>                                     - each Meth element is the same as an actual API request message
  <MName>APIRole.New.base</MName>
  <PList>
    <Prm>
      <Name>RoleName</Name>
      <Val>
        <![CDATA[]]>
      </Val>
    </Prm>
    <Prm>
      <Name>RoleDescrip</Name>
      <Val>
        <![CDATA[]]>
      </Val>
    </Prm>
  </PList>
</Meth>
<RList>                                    - a list of one or more results returned by the test message
  <Result>
    <RName>APIRole.New.base_DEFAULT</RName>
    <ExpRCode>OK</ExpRCode>                - the expected Result Code
    <ExpDType>TEXT</ExpDType>              - the expected Data Type
    <ExpRVal></ExpRVal>                    - the expected Result Value
    <ValMatch></ValMatch>                  - flag value indicating if the returned value matches the expected value
    <VarName></VarName>                    - a name for the result value returned by the test message, which can be used as if it were
  </Result>                                a template placeholder in test messages run later in the test batch.
```


© 2021 Alan Rahn All Rights Reserved

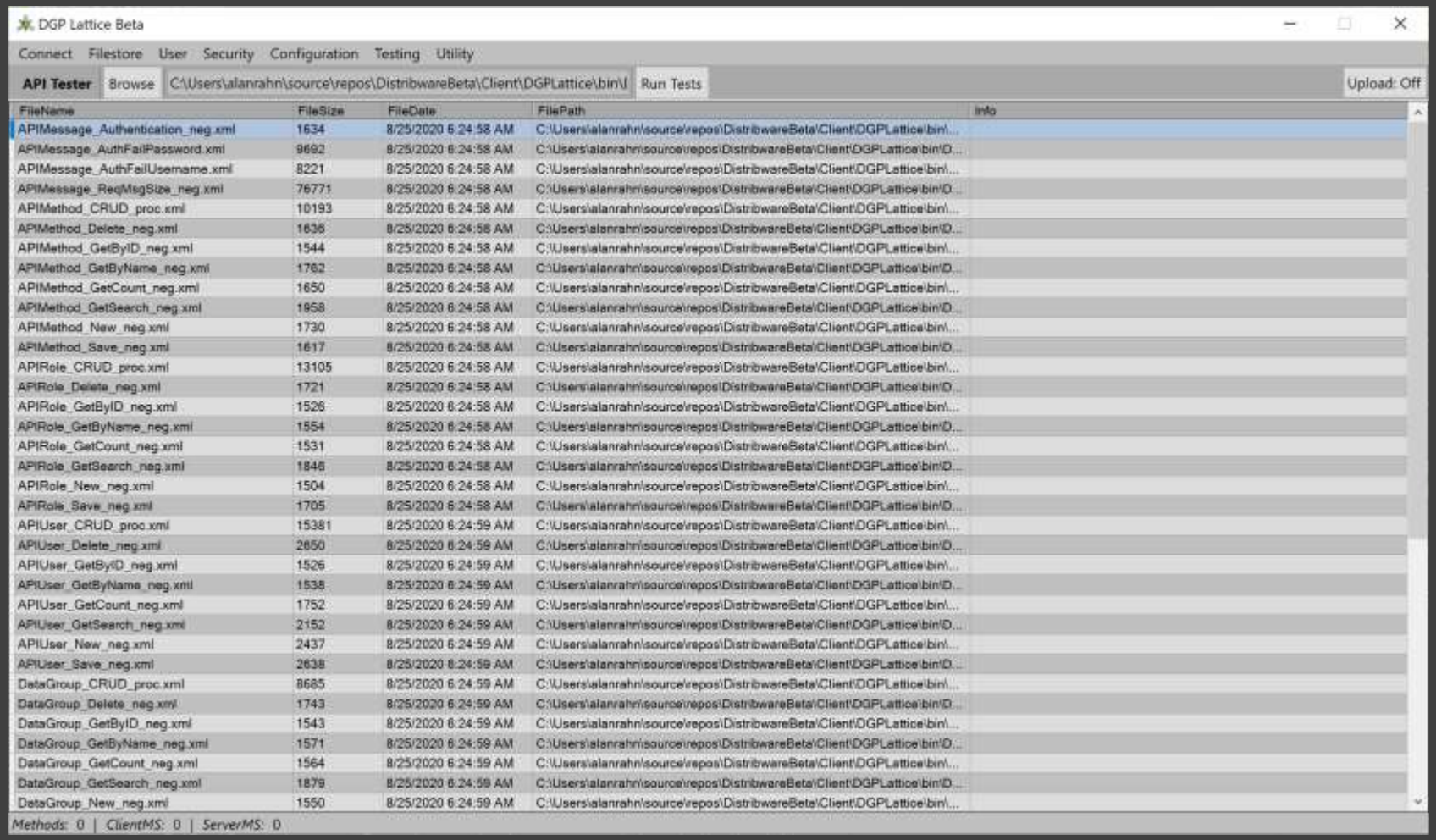
```
    </RList>
  </TMsg>
</TestBatch>
```

The first step when using the API Tester test harness is to click the Browse button and select a folder containing DGP test files.



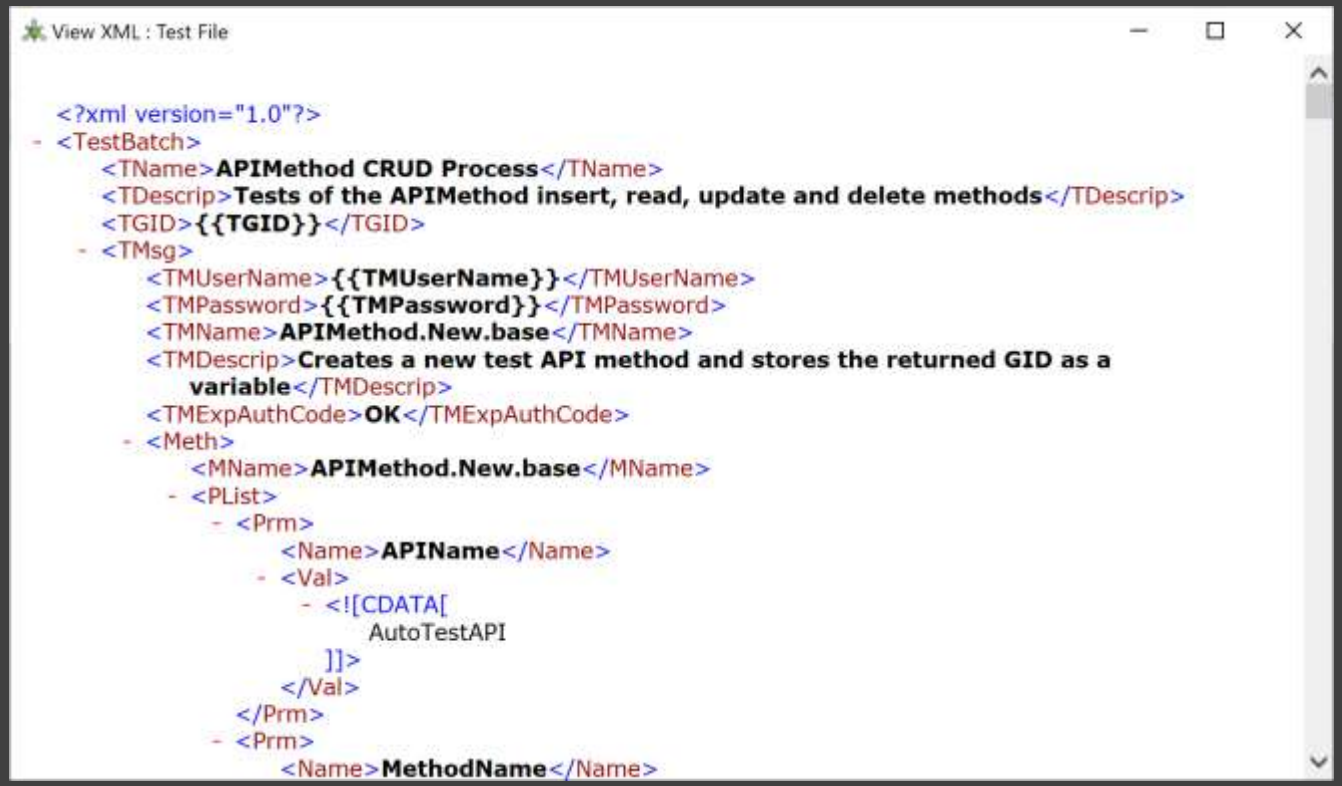
© 2021 Alan Rahn *All Rights Reserved*

Whichever folder is chosen will select all DGP test files in that folder plus all subfolders below it, recursively. For example, to run a full regression of all tests of all API methods, select the top "TestFiles" folder. To select fewer test files, drill down below the root folder to the desired subfolder. Note: all test files for DGP Lattice are included as part of the open source project.



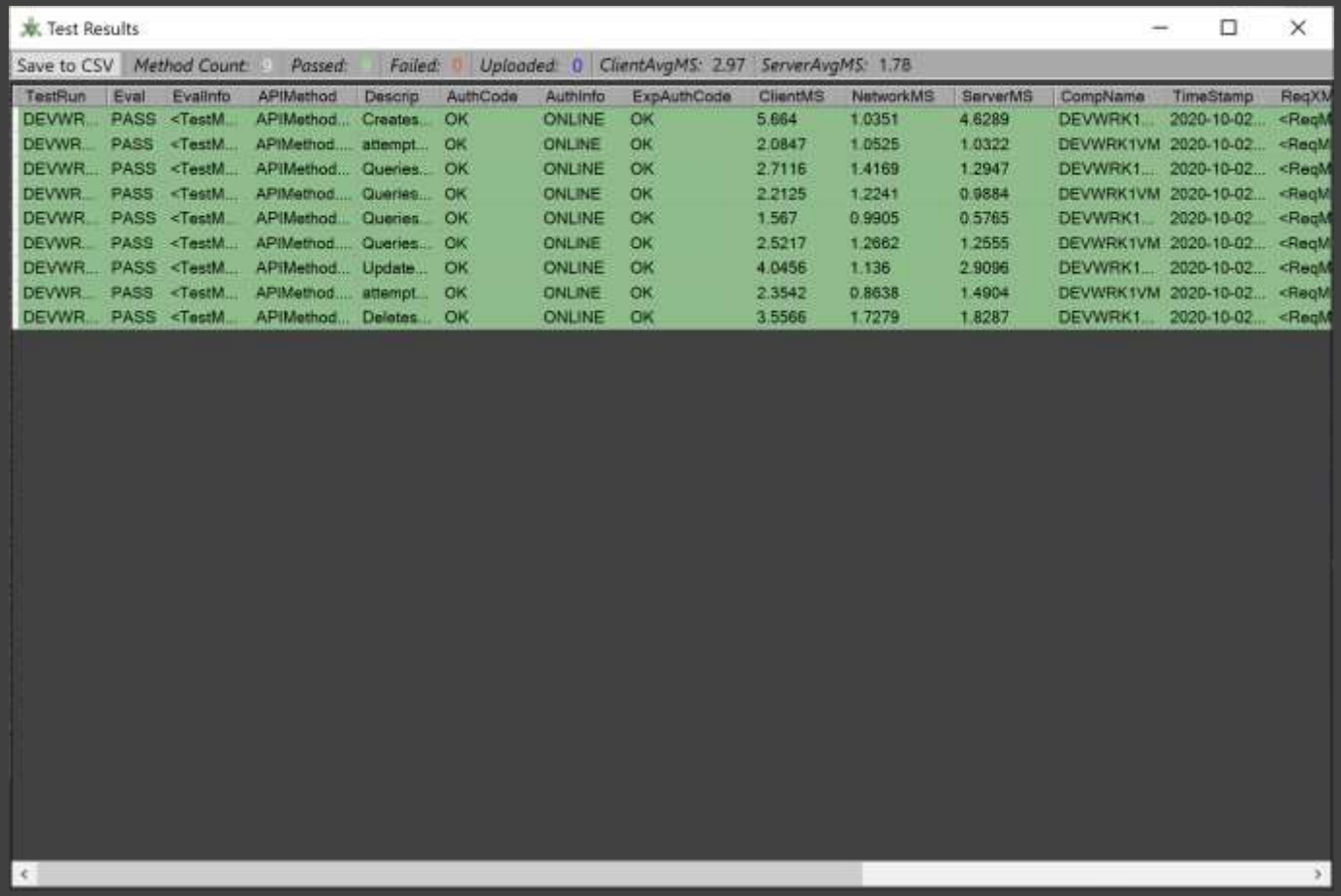
© 2021 Alan Rahn *All Rights Reserved*

This example above shows all of the test files in the TestFiles folder and all of its subfolders, recursively. The context menu in the test file grid only has a single option: View File. This option allows users to view a test file, but the content is read-only and cannot be edited.



© 2021 Alan Rahn *All Rights Reserved*

Selecting a single test file to run will include more data in each row of the Test Results form, compared to running multiple test files as a batch. This is done to reduce the size of the total data stored in the results form. Otherwise, select as many test files to run as desired using CTRL + click, or select all test files using SHIFT + click, etc.
When the selection is complete, click the Run Tests button. All the methods in all the selected tests will be run in sequential order. In the example below, a single test file (APIMethod_CRUD_proc.xml) was selected and run.



© 2021 Alan Rahn *All Rights Reserved*

The results of the single APIMethod_CRUD_proc.xml test file are shown above. The test file contained 9 test messages.

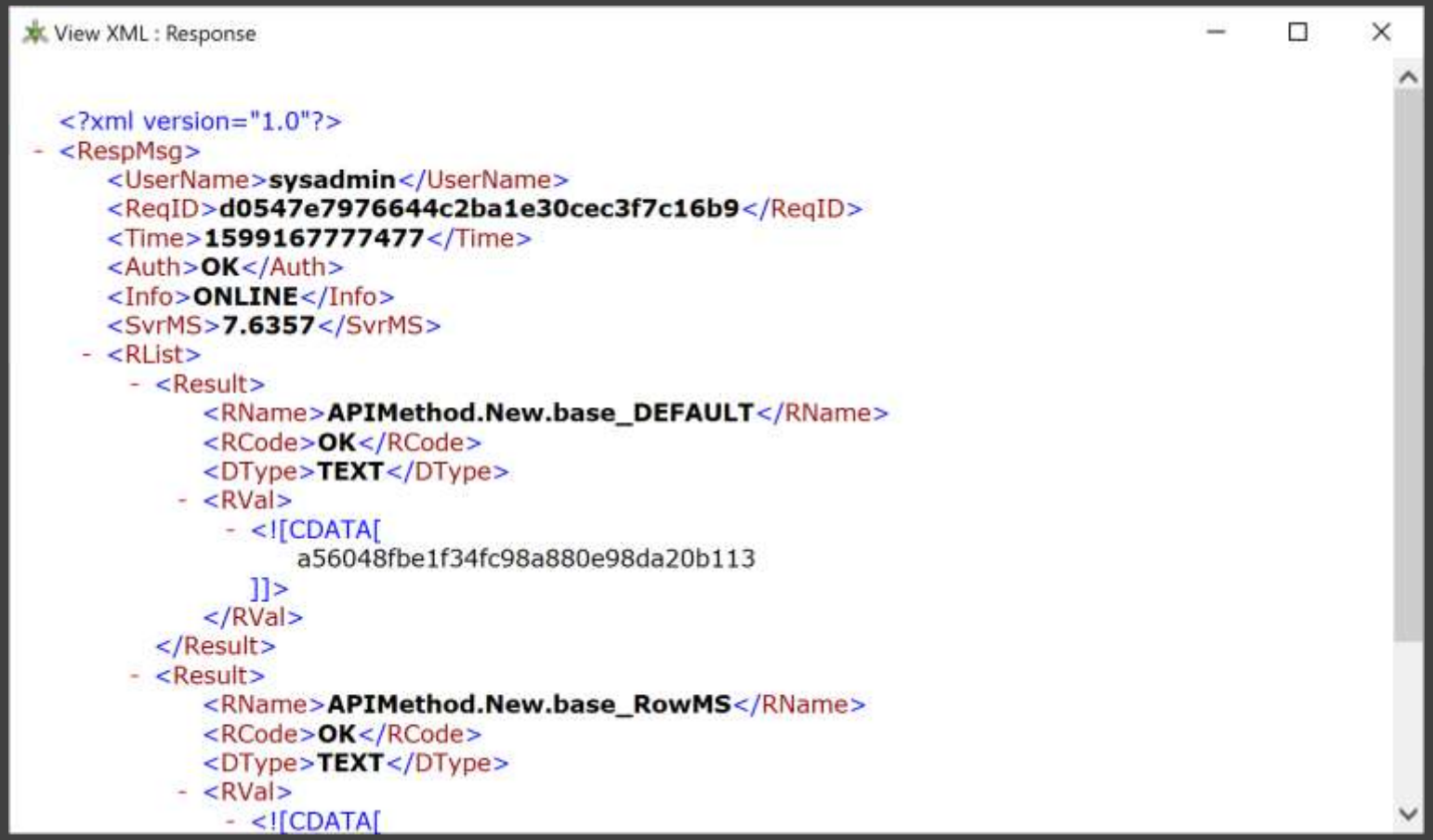
Since only a single test file was run, the context menu for each row contains 3 options: View Eval Info, View Request Message, and View Response Message.

The evaluation info shows all of the values that were expected by the test, compared to the actual values returned when the test was run. If the evaluation failed, the next step is to look at the request message sent to the web service and the response message returned by the web service to look for any problems or errors.

Note: the request and response messages are actually xml fragments, but in order to get the web browser control to display the XML correctly, an XML header was added. Sending an API request message with an XML header would be rejected by the web service.

© 2021 Alan Rahn *All Rights Reserved*

© 2021 Alan Rahn *All Rights Reserved*



© 2021 Alan Rahn *All Rights Reserved*

TestRun	Eval	EvalInfo	APIMethod	Descrip	AuthCode	AuthInfo	ExpAuthCode	ClientMS	NetworkMS	ServerMS	CompName	TimeStamp	Rec
DEVWR...	PASS	<TestM...	TestReplic...	Creates...	OK	ONLINE	OK	28.6021	1.746	26.8561	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	TestReplic...	Creates...	OK	ONLINE	OK	25.8259	0.91789999...	24.908	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	Test.EchoT...	test of t...	OK	ONLINE	OK	0.9743	0.8413	0.133	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	Test.Loggi...	test log...	OK	ONLINE	OK	4.0722	0.95000000...	3.1222	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	Test.Excep...	test of t...	OK	ONLINE	OK	2.8987	0.9488	1.9499	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	APIUser.Ch...	check f...	OK	ONLINE	OK	1.4513	1.012	0.4393	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	APIUser.N...	Creates...	OK	ONLINE	OK	14.605	0.95690000...	13.6481	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	UserSelf.L...	get web...	EXPIRED	ONLINE	EXPIRED	40.1853	0.99050000...	39.1948	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	UserSelf.C...	updates...	EXPIRED	ONLINE	EXPIRED	36.3634	0.9435	35.4199	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	UserSelf.G...	queries ...	OK	ONLINE	OK	14.1024	0.99200000...	13.1104	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	UserSelf.G...	Query f...	OK	ONLINE	OK	0.8501	0.7539	0.0962	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	UserSelf.G...	Query f...	OK	ONLINE	OK	12.9708	0.9808	11.99	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	UserSelf.S...	empty i...	OK	ONLINE	OK	25.614	1.2595	24.3545	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	APIUser.De...	delete t...	OK	ONLINE	OK	24.8452	0.92000000...	23.9252	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser....	empty i...	OK	ONLINE	OK	0.9748	0.842	0.1328	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser.R...	missing ...	OK	ONLINE	OK	0.8246	0.7259	0.0987	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser....	empty i...	OK	ONLINE	OK	0.9567	0.8221	0.1346	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser.G...	missing ...	OK	ONLINE	OK	0.9605	0.848	0.1125	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser....	empty i...	OK	ONLINE	OK	0.8486	0.68	0.1686	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser.G...	missing ...	OK	ONLINE	OK	0.7891	0.6834	0.1057	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	APIRole.N...	Creates...	OK	ONLINE	OK	2.2184	0.8592	1.3592	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	APIUser.Ne...	Creates...	OK	ONLINE	OK	14.8227	0.8977	13.925	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser....	Query f...	OK	ONLINE	OK	1.8286	0.8961	0.9325	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser.A...	assign t...	OK	ONLINE	OK	2.2916	1.0379	1.2537	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser....	attempt ...	OK	ONLINE	OK	1.6748	0.8878	0.787	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser.G...	Query f...	OK	ONLINE	OK	1.8452	0.8237	1.0215	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser....	remove...	OK	ONLINE	OK	2.185	0.8571	1.3279	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	APIUser.De...	delete t...	OK	ONLINE	OK	25.3553	0.92500000...	24.4303	DEVWRK1VM	2020-10-02...	
DEVWR...	PASS	<TestM...	APIRole.D...	Deletes...	OK	ONLINE	OK	2.8438	1.1803	1.6635	DEVWRK1...	2020-10-02...	
DEVWR...	PASS	<TestM...	RoleUser.A...	empty i...	OK	ONLINE	OK	0.9775	0.8493	0.1282	DEVWRK1VM	2020-10-02...	

© 2021 Alan Rahn All Rights Reserved

When many test files are selected and run as a batch (for example, as a full regression test of all API methods), the context menu items View Request Message and View Response Message are disabled, since that data is not included in the result table records to help prevent the data in the grid from growing too large. If that ever becomes a problem, the easiest solution is to test batches of files rather than all of them at once.

Finally, the test results can be saved to the SysMetrics database of the location being tested by clicking the Upload button at the top right of the test harness ***prior to running the tests***. If you forget to do that, just click the Upload button and run the tests again.

Finally, the test results displayed in the result form can be saved to a local CSV file by clicking the Save to CSV button at the top left of the screen. This opens a dialog to select where to store the .csv file.

Manual Testing

The full regression tests of (almost) all API methods in a system covers the majority of the testing workload. However, some data-driven process methods cannot be tested by the API Tester for a variety of reasons. Also, the UI's themselves must still be tested as well. These types of tests are manual by default, and are standardized by using manual test scripts that are followed by the testers each time they are "run" for the sake of consistency.

The easiest way to accomplish this manual testing is to use tabular templates for each test script (Excel, Word tables, etc.) that contain a row for each step of the test. Each row in turn contains a name for the step, a description of inputs and expected outputs, a place to document the observed results, and the evaluation of the step (pass / fail). A tester running a manual test would create a copy of the template, and then perform each step of the test like a checklist, documenting the results of each step as they go. When finished, the tester would then save the filled-in checklist with a name that includes the name of the template and the date the manual test was run into some type of shared storage in order to verify that the manual tests were run and to preserve the results of the tests.

© 2021 Alan Rahn All Rights Reserved

Manual Tests:

- Recover methods (recover edited version, recover deleted record)
- File upload
- File download
- Replicate methods
- GetSrcCount methods
- GetDestCount methods

In general, Recover methods and all of the API methods used by the replication processes must be tested manually after proper configuration in a given environment.

© 2021 Alan Rahn *All Rights Reserved*