Testing Overview

One of the most important features needed to improve the quality of software systems is continuous testing. The DGP design contains a number of innovative capabilities, and one of them is the end-to-end unit testing and performance monitoring that is built into each API method in the platform.

The DGP architecture deliberately consolidates all possible functionality in a system into its web service API's, which greatly improves performance, reusability, simplifies maintenance and deployments, and increases the security of the system. It also makes it much easier to test all of the functionality in a system by integrating end-to-end unit tests into each API method. That, combined with the built-in test harness application allows full regression tests of all API methods in a system to be run very easily.

Testing Correct Functionality

The two-way communication of data from the client app to the server in the request message, and from the server to the client app in the response message depends on the use of message-based RPC style API's, along with the single-argument front controller pattern for the API controllers. Message-based API methods enable the sharing of multiple different types of additional data between the client app and the server API beyond the usual input parameters and method return value of standard method calls.

Within the web services, every API mapper method performs an evaluation of the result returned by the internal library method it calls. Based on this evaluation, it then assigns one of the ResultCode values (OK, Empty, Error, and Exception). The result codes, result info, method return values and server performance data are then returned to the client app as part of every API response message. This allows a DGP system to verify the security, correct functionality and also measure the end-to-end performance of every API method in that system, quite literally every time that method is called by each user of the system. This also allows the test and performance data to be collected during the normal use of every production DGP system, without adversely affecting the performance of that system. No other platform provides similar continuous, universal testing and monitoring functionality.

A security system role is used by the DGPDrive client application to enable the testing and monitor data collected in each client app to be saved to the databases of the connected location. This controls how many and which specific users are saving this type of data by their membership in the RemoteMonitor role.

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
          <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>
```

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 by the API Tester test harness app to create correctly formatted API request messages. 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 standard API request message because each test is run one at a time. Test API method calls cannot be batched like in a normal API request message and must be called sequentially in order 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 run as a series of sequential steps, calling different API methods, similar to what a user would do when using an application, etc. For example, 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.

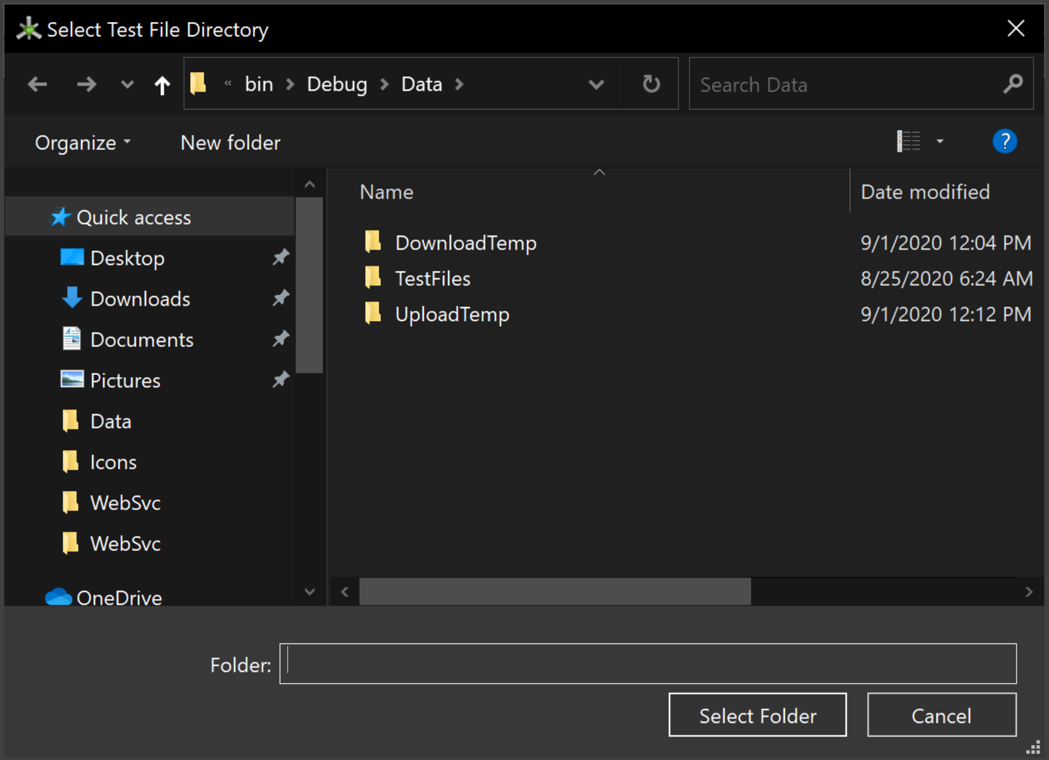
The tests themselves can be thought of as “end-to-end unit tests” thanks to some features built into the DGP web service API’s. As mentioned previously, the last step in the mapper methods of the web services analyze the method results returned by the internal method called and assign one of the ResultCode values (OK, Empty, Error, Exception) in the API response message. The ResultCode is similar to a unit test assertion, and that functionality is what makes the API Tester test harness application itself possible. Rather than have the client app contain logic to test the results of each method call, that logic is centralized within each API Mapper method. The two-way communication capabilities of the message-based API’s are then used to return the results of the “unit test”.

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>
    <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>
```

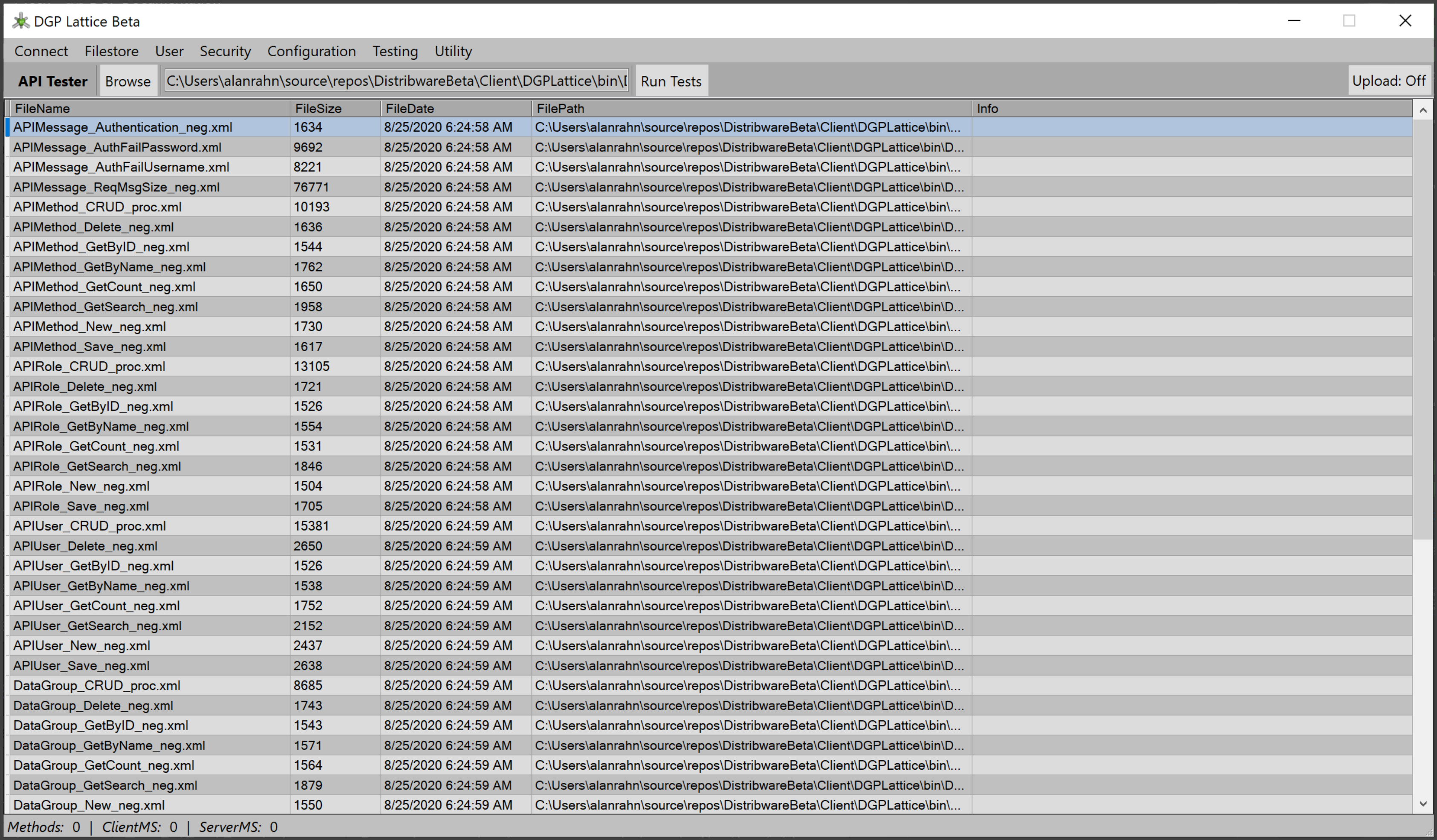
```
            <![CDATA[]]>
          </Val>
        </Prm>
        <Prm>
          <Name>RoleDescrip</Name>
          <Val>
            <![CDATA[]]>
          </Val>
        </Prm>
      </PList>
    </Meth>
    <RList>                             - a list of one or more results returned by the test 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.
    </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.



Name	Date modified
DownloadTemp	9/1/2020 12:04 PM
TestFiles	8/25/2020 6:24 AM
UploadTemp	9/1/2020 12:12 PM

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 a subset of fewer test files, drill down below the root folder to the desired subfolder. Note: all test files for DGPDrive are included as part of the open source project.

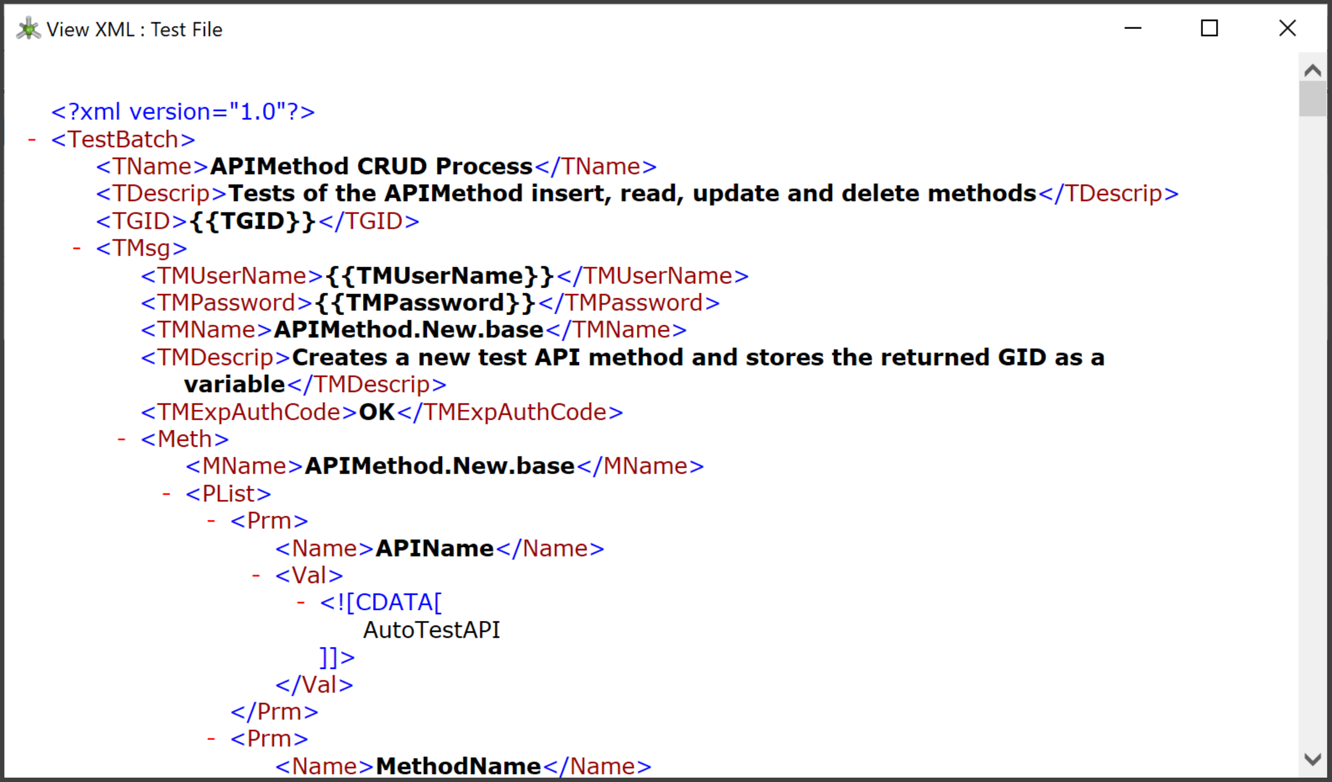


DGP Lattice Beta
Connect Filestore User Security Configuration Testing Utility
API Tester
Browse
C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\[
Run Tests
Upload: Off

FileName	FileSize	FileDate	FilePath	Info
APIMessage_Authentication_neg.xml	1634	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIMessage_AuthFailPassword.xml	9692	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIMessage_AuthFailUsername.xml	8221	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIMessage_ReqMsgSize_neg.xml	76771	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIMethod_CRUD_proc.xml	10193	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIMethod_Delete_neg.xml	1636	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIMethod_GetByID_neg.xml	1544	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIMethod_GetByName_neg.xml	1762	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIMethod_GetCount_neg.xml	1650	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIMethod_GetSearch_neg.xml	1958	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIMethod_New_neg.xml	1730	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIMethod_Save_neg.xml	1617	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIRole_CRUD_proc.xml	13105	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIRole_Delete_neg.xml	1721	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIRole_GetByID_neg.xml	1526	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIRole_GetByName_neg.xml	1554	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIRole_GetCount_neg.xml	1531	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIRole_GetSearch_neg.xml	1846	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIRole_New_neg.xml	1504	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIRole_Save_neg.xml	1705	8/25/2020 6:24:58 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIUser_CRUD_proc.xml	15381	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIUser_Delete_neg.xml	2650	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIUser_GetByID_neg.xml	1526	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIUser_GetByName_neg.xml	1538	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIUser_GetCount_neg.xml	1752	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIUser_GetSearch_neg.xml	2152	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
APIUser_New_neg.xml	2437	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
APIUser_Save_neg.xml	2638	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
DataGroup_CRUD_proc.xml	8685	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
DataGroup_Delete_neg.xml	1743	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
DataGroup_GetByID_neg.xml	1543	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
DataGroup_GetByName_neg.xml	1571	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
DataGroup_GetCount_neg.xml	1564	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	
DataGroup_GetSearch_neg.xml	1879	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\D...	
DataGroup_New_neg.xml	1550	8/25/2020 6:24:59 AM	C:\Users\alanrahn\source\repos\DistribwareBeta\Client\DGPLattice\bin\...	

Methods: 0 | ClientMS: 0 | ServerMS: 0

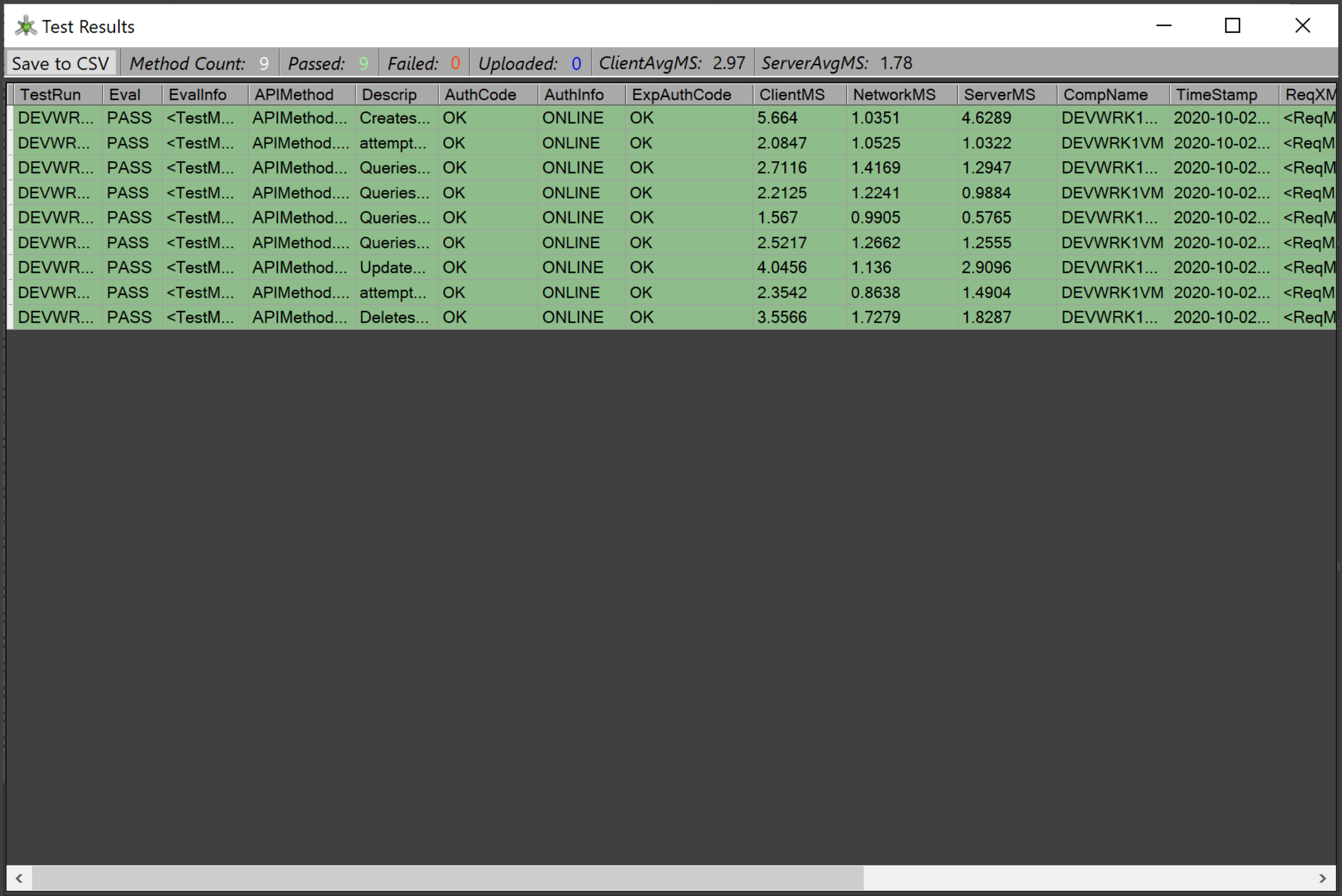
This example above shows all of the test files in the TestFiles folder and all of its subfolders, populated recursively. The context menu in the test file grid only has a single option: View File. This option allows users to view a test file, but the content is read-only and cannot be edited.



```
  <?xml version="1.0"?>
- <TestBatch>
      <TName>APIMethod CRUD Process</TName>
      <TDescrip>Tests of the APIMethod insert, read, update and delete methods</TDescrip>
      <TGID>{{TGID}}</TGID>
    - <TMsg>
          <TMUserName>{{TMUserName}}</TMUserName>
          <TMPassword>{{TMPassword}}</TMPassword>
          <TMName>APIMethod.New.base</TMName>
          <TMDescrip>Creates a new test API method and stores the returned GID as a
              variable</TMDescrip>
          <TMExpAuthCode>OK</TMExpAuthCode>
        - <Meth>
              <MName>APIMethod.New.base</MName>
            - <PList>
                - <Prm>
                      <Name>APIName</Name>
                    - <Val>
                        - <![CDATA[
                              AutoTestAPI
                          ]]>
                      </Val>
                  </Prm>
                - <Prm>
                      <Name>MethodName</Name>
```

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 amount of 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.

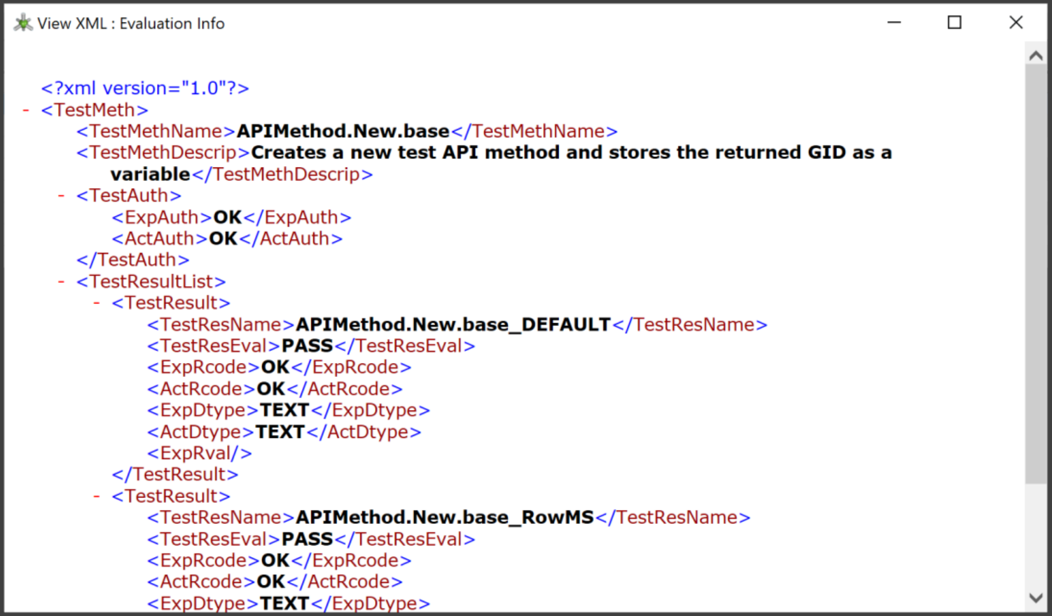


Method Count: 9 | *Passed:* 9 | *Failed:* 0 | *Uploaded:* 0 | *ClientAvgMS:* 2.97 | *ServerAvgMS:* 1.78

TestRun	Eval	EvalInfo	APIMethod	Descrip	AuthCode	AuthInfo	ExpAuthCode	ClientMS	NetworkMS	ServerMS	CompName	TimeStamp	ReqXM
DEVWR...	PASS	<TestM...	APIMethod...	Creates...	OK	ONLINE	OK	5.664	1.0351	4.6289	DEVWRK1...	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod....	attempt...	OK	ONLINE	OK	2.0847	1.0525	1.0322	DEVWRK1VM	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod...	Queries...	OK	ONLINE	OK	2.7116	1.4169	1.2947	DEVWRK1...	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod....	Queries...	OK	ONLINE	OK	2.2125	1.2241	0.9884	DEVWRK1VM	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod...	Queries...	OK	ONLINE	OK	1.567	0.9905	0.5765	DEVWRK1...	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod....	Queries...	OK	ONLINE	OK	2.5217	1.2662	1.2555	DEVWRK1VM	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod...	Update...	OK	ONLINE	OK	4.0456	1.136	2.9096	DEVWRK1...	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod....	attempt...	OK	ONLINE	OK	2.3542	0.8638	1.4904	DEVWRK1VM	2020-10-02...	<ReqM
DEVWR...	PASS	<TestM...	APIMethod...	Deletes...	OK	ONLINE	OK	3.5566	1.7279	1.8287	DEVWRK1...	2020-10-02...	<ReqM

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

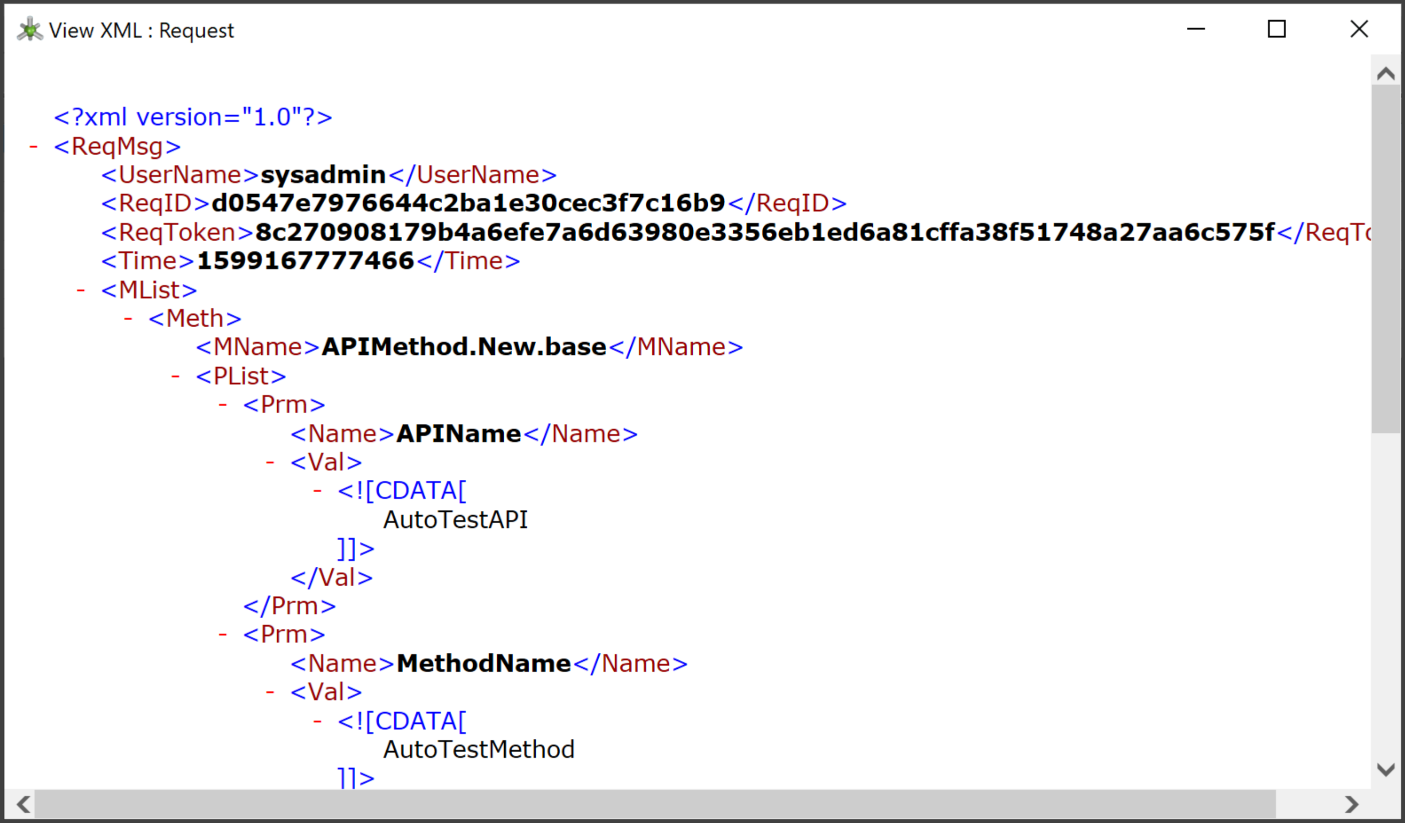
Since only a single test file was run, the context menu for each row contains 3 options: View Eval Info, View Request Message, and View Response Message.



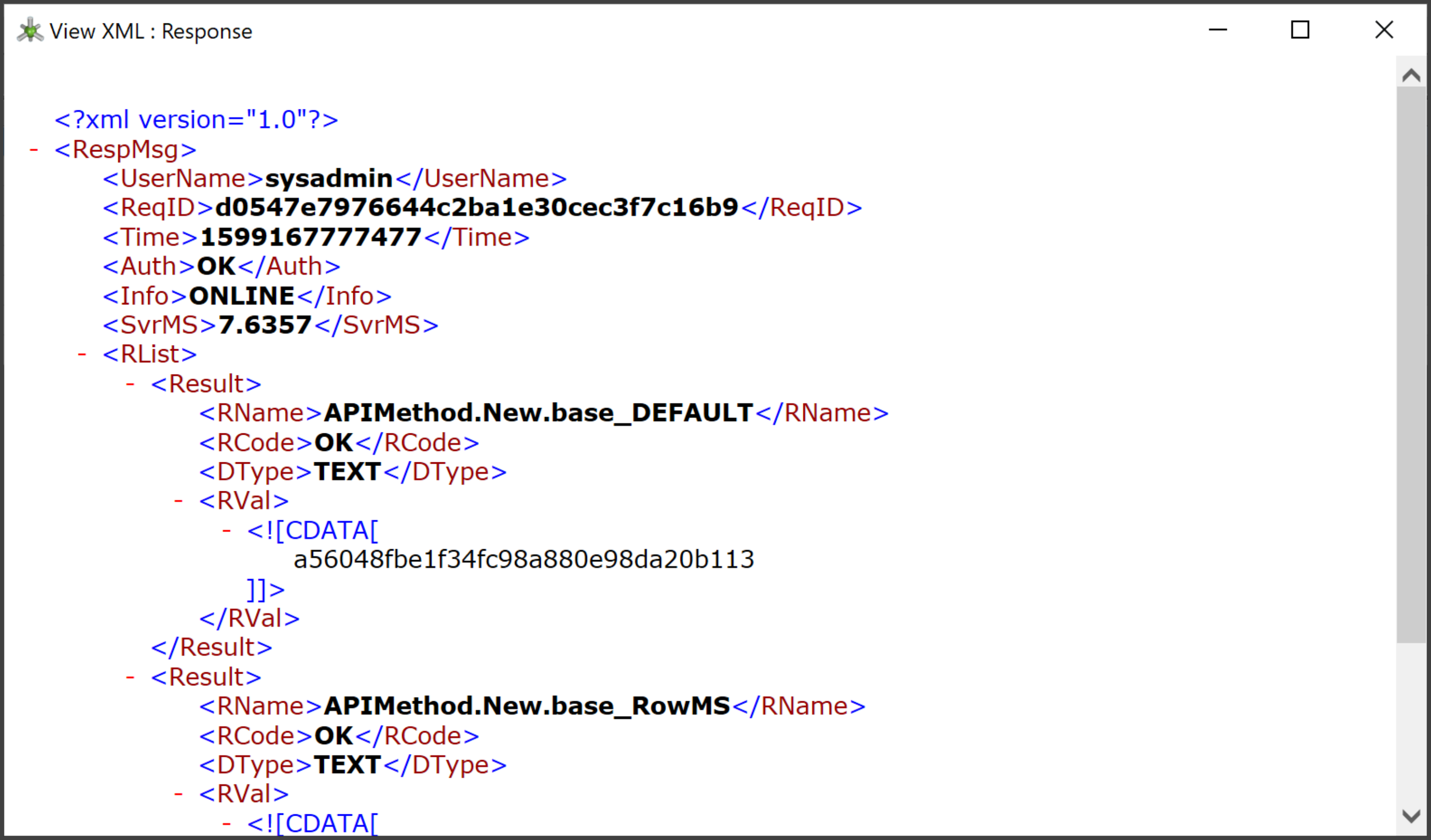
```
  <?xml version="1.0"?>
- <TestMeth>
     <TestMethName>APIMethod.New.base</TestMethName>
     <TestMethDescrip>Creates a new test API method and stores the returned GID as a
        variable</TestMethDescrip>
   - <TestAuth>
        <ExpAuth>OK</ExpAuth>
        <ActAuth>OK</ActAuth>
     </TestAuth>
   - <TestResultList>
      - <TestResult>
           <TestResName>APIMethod.New.base_DEFAULT</TestResName>
           <TestResEval>PASS</TestResEval>
           <ExpRcode>OK</ExpRcode>
           <ActRcode>OK</ActRcode>
           <ExpDtype>TEXT</ExpDtype>
           <ActDtype>TEXT</ActDtype>
           <ExpRval/>
        </TestResult>
      - <TestResult>
           <TestResName>APIMethod.New.base_RowMS</TestResName>
           <TestResEval>PASS</TestResEval>
           <ExpRcode>OK</ExpRcode>
           <ActRcode>OK</ActRcode>
           <ExpDtype>TEXT</ExpDtype>
```

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.

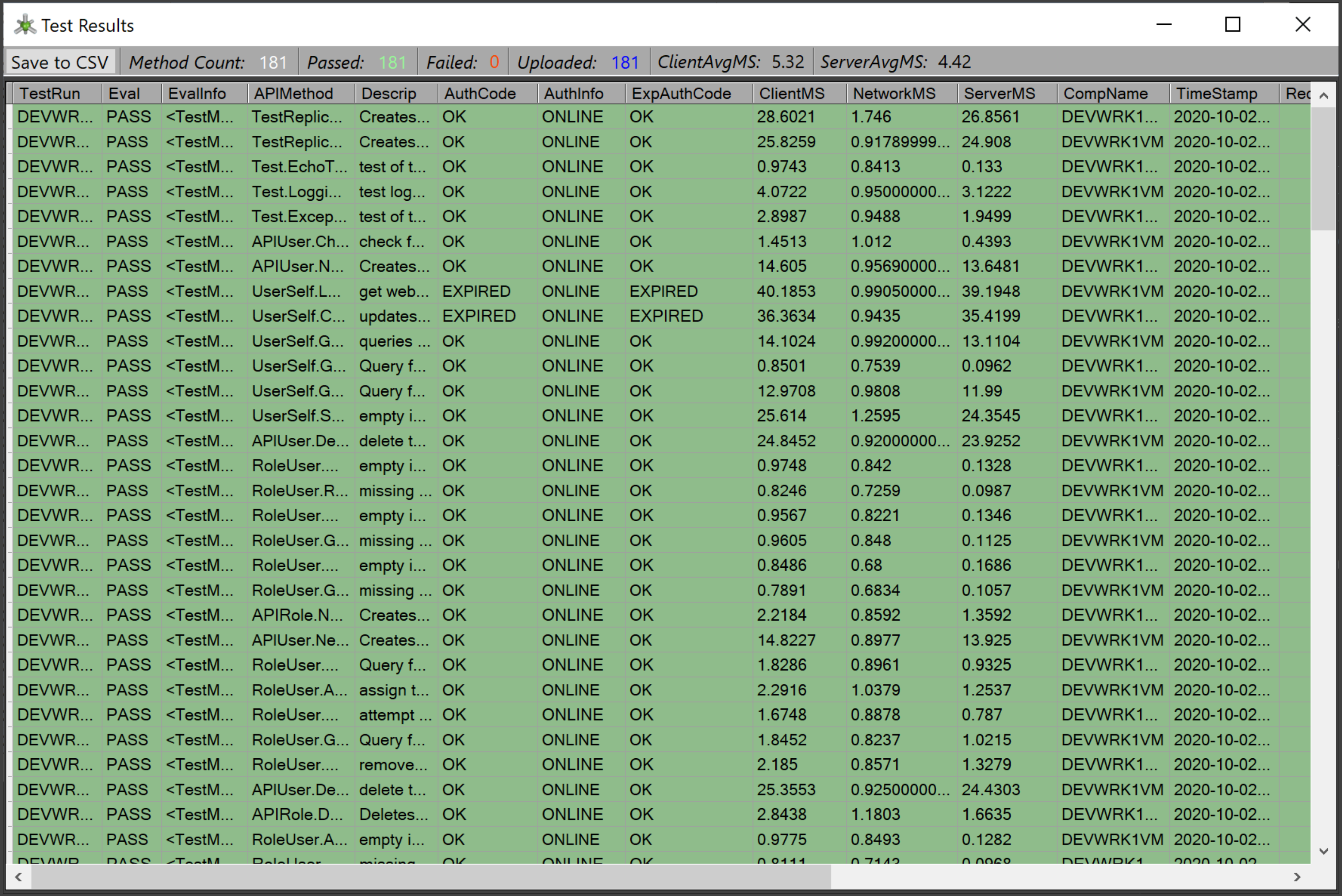
Note: the request and response messages are actually xml fragments, but in order to get the web browser control to display the XML correctly, an XML header was added. Sending an API request message with an XML header would be rejected by the web service, because it only accepts XML fragments for security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



```
  <?xml version="1.0"?>
- <ReqMsg>
      <UserName>sysadmin</UserName>
      <ReqID>d0547e7976644c2ba1e30cec3f7c16b9</ReqID>
      <ReqToken>8c270908179b4a6efe7a6d63980e3356eb1ed6a81cffa38f51748a27aa6c575f</ReqTo
      <Time>1599167777466</Time>
    - <MList>
        - <Meth>
              <MName>APIMethod.New.base</MName>
            - <PList>
                - <Prm>
                      <Name>APIName</Name>
                    - <Val>
                        - <![CDATA[
                              AutoTestAPI
                          ]]>
                      </Val>
                  </Prm>
                - <Prm>
                      <Name>MethodName</Name>
                    - <Val>
                        - <![CDATA[
                              AutoTestMethod
                          ]]>
```



```
  <?xml version="1.0"?>
- <RespMsg>
      <UserName>sysadmin</UserName>
      <ReqID>d0547e7976644c2ba1e30cec3f7c16b9</ReqID>
      <Time>1599167777477</Time>
      <Auth>OK</Auth>
      <Info>ONLINE</Info>
      <SvrMS>7.6357</SvrMS>
    - <RList>
        - <Result>
              <RName>APIMethod.New.base_DEFAULT</RName>
              <RCode>OK</RCode>
              <DType>TEXT</DType>
            - <RVal>
                - <![CDATA[
                      a56048fbe1f34fc98a880e98da20b113
                  ]]>
              </RVal>
          </Result>
        - <Result>
              <RName>APIMethod.New.base_RowMS</RName>
              <RCode>OK</RCode>
              <DType>TEXT</DType>
            - <RVal>
                - <![CDATA[
```



Method Count: 181 | *Passed:* 181 | *Failed:* 0 | *Uploaded:* 181 | *ClientAvgMS:* 5.32 | *ServerAvgMS:* 4.42

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...	
DEVWR...	PASS	<TestM...	RoleUser....	missing ...	OK	ONLINE	OK	0.8111	0.7143	0.0968	DEVWRK1...	2020-10-02...	

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 DataTable use as a binding source by the DataGrid from growing too large. If that ever becomes a problem, the easiest solution is to test batches of files (subsets) 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. Running a full regression of all of the API methods in a DGPDrive system typically only takes a few seconds when running localhost, but can take a few minutes if network latency is significant.

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 (UI + API)

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 functionality of 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 some type of documentation template 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.

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.

Load/Stress Testing

The ability to perform load/stress testing of a system in one of the lower environments (typically the QA environment) is very important.

Long term synthetic client application load.

Long term server performance and resource utilization measurements.

Reliability Testing

Reliability testing is primarily concerned with the redundant infrastructure used to provide 100% system availability. The basic level of redundancy in DGP systems consist of multiple locations with data replicated between them, which can be either individual computers at a small scale, or sets of servers connected by a LAN at a larger scale. These locations, in turn, are connected together by a network of the appropriate scale (LAN or WAN, etc.).

- Location flagged as offline (generally for maintenance)

- Client application failover from one location to another
    - Potential loss of data
    - Duplication of lost data
- Automated process pause and recovery
    - Data replication pause and recovery
    - Data verification pause and recovery
    - General process pause and recovery

Testing Correct Data (DGP Replication)

The data replication process is itself very reliable, and has multiple safeguards built into the logic of the automated process. However, no single process can ever be perfectly foolproof all by itself. For this reason, another automated process is designed specifically to scan all of the replicated data at each destination, comparing it to the matching data at the source. This automation, just like data replication, is designed to be run continuously as a background process, scanning and rescanning the data – looking for any errors, over and over again.

The advantage of this approach is that it detects errors of all kinds, which can occur at any time (not just during data replication). For example, these tests can detect and correct damage to the data caused by mistakes made by system admins, and so on. Given the immutable append-only conventions used for replica data, there are 3 main types of errors that can occur:

- Missing records – automatically replaced by the data verification process, and logged
- Duplicate records – automatically flagged to be ignored by the system by the data verification process, and logged
- Inconsistent record data – flagged, logged, plus a notification message is sent to system admins