Testing

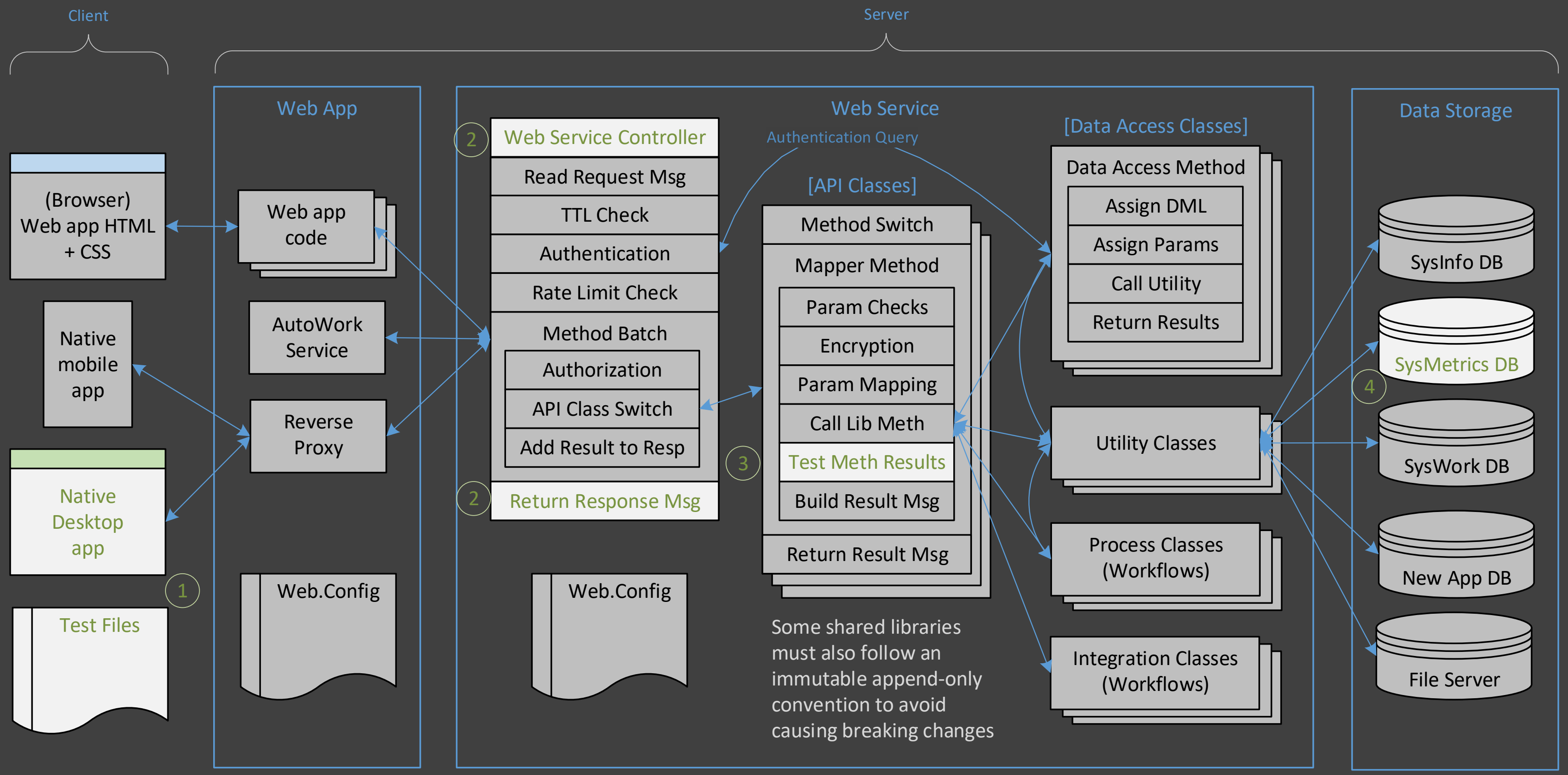
Testing is an essential part of a DGP system in order to prove that all of its functionality is working correctly and performing well. It is also a necessary requirement for the CI/CD/CT capabilities that greatly simplify frequent deployments, testing and rollbacks. Compared to standard method calls, 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. In other words, every API method has built-in unit testing, those tests are run every time an API method is called, and the results of those tests are returned as part of every API response message – along with the server performance metrics.

This enables the built-in API Tester test harness 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. Some process methods require dynamic state and configuration data at runtime, and are therefore not able to be tested by the API Tester. Those methods must be tested manually using the AutoWork Test harness instead. The test results from any test run are able to be saved for analysis over time. The beta version of the Lattice API Tester test harness runs a full regression of 220 API method calls (both positive and negative tests) in under 3 seconds.

The API Tester test files cover both primary path and alternate paths through the code of each API method. The API method calls in a test file can be linked together so that the results of one method call can be used as an input in a later method call within a test file. In addition, all tests of the API methods measure the end-to-end performance of each method call, while the server-side performance is returned as part of each response message.

The AutoWork windows service running as a console app is used to verify the correct functionality of the mechanisms to claim, run, and reschedule automated processes, as well as the performance of the various steps in those processes stored in AutoWorkLog records. Running the AutoWork service as a console app is also useful when debugging the execution of the automated processes themselves.

Distributed Grid Platform (DGP): Testing Design



Client

(Browser) Web app HTML + CSS

Native mobile app

Native Desktop app

1

Test Files

Server

Web App

Web app code

AutoWork Service

Reverse Proxy

Web.Config

Web Service

2 Web Service Controller

Read Request Msg

TTL Check

Authentication

Rate Limit Check

Method Batch

Authorization

API Class Switch

Add Result to Resp

2 Return Response Msg

Web.Config

Authentication Query

[API Classes]

Method Switch

Mapper Method

Param Checks

Encryption

Param Mapping

Call Lib Meth

3 Test Meth Results

Build Result Msg

Return Result Msg

Some shared libraries must also follow an immutable append-only convention to avoid causing breaking changes

[Data Access Classes]

Data Access Method

Assign DML

Assign Params

Call Utility

Return Results

Utility Classes

Process Classes (Workflows)

Integration Classes (Workflows)

Data Storage

SysInfo DB

4 SysMetrics DB

SysWork DB

New App DB

File Server

1. The native desktop app contains test harness UI's that are used to test all the API methods of all API classes. The tests are basically API request messages that have been saved as a template, along with other info useful to the test harness logic. These test files contain both positive and negative tests of each API method. The test harness allows method calls in a test file to be chained together sequentially, where the data returned by one method can become an input value for subsequent method calls.

2. The web service controller methods measure the elapsed time for each API request, and returns the performance metric in each response message.

3. Each API mapper method contains logic to test the results returned by the call to the internal library method. These tests are then assigned a category value (result code) indicating the success or failure of the internal library method call, and the type of failure if necessary. The test harness then compares the actual result code returned to the expected result code (positive or negative) to determine if the end-to-end test of the API method was successful.

4. The test results can be saved for long term trend analysis in the SysMetrics database, and/or exported to a CSV file.

Testing Verification

1. *Multiple test files are created for every web service API method in a system. Some test the primary path through the code of the method, while others test alternate (error) paths through the code. The collection of test files should provide 100% code coverage for each method.*
2. *Each test file can contain many API method calls, and chain them together such that the results returned from one method call can be used as an input into other method calls.*
3. *The API Tester test harness runs all selected test files, creating API request messages from the content of each test file, and using those request messages to call the API methods of the location it is connected to.*
4. *The testing, detection, repair and monitoring tools are themselves free and open-source, and are included as part of DGP. This enables the correct functionality of the tests and test harness logic to be independently verified by anyone that has access to the source code, stepping through the code as it runs.*