Main Form Overview

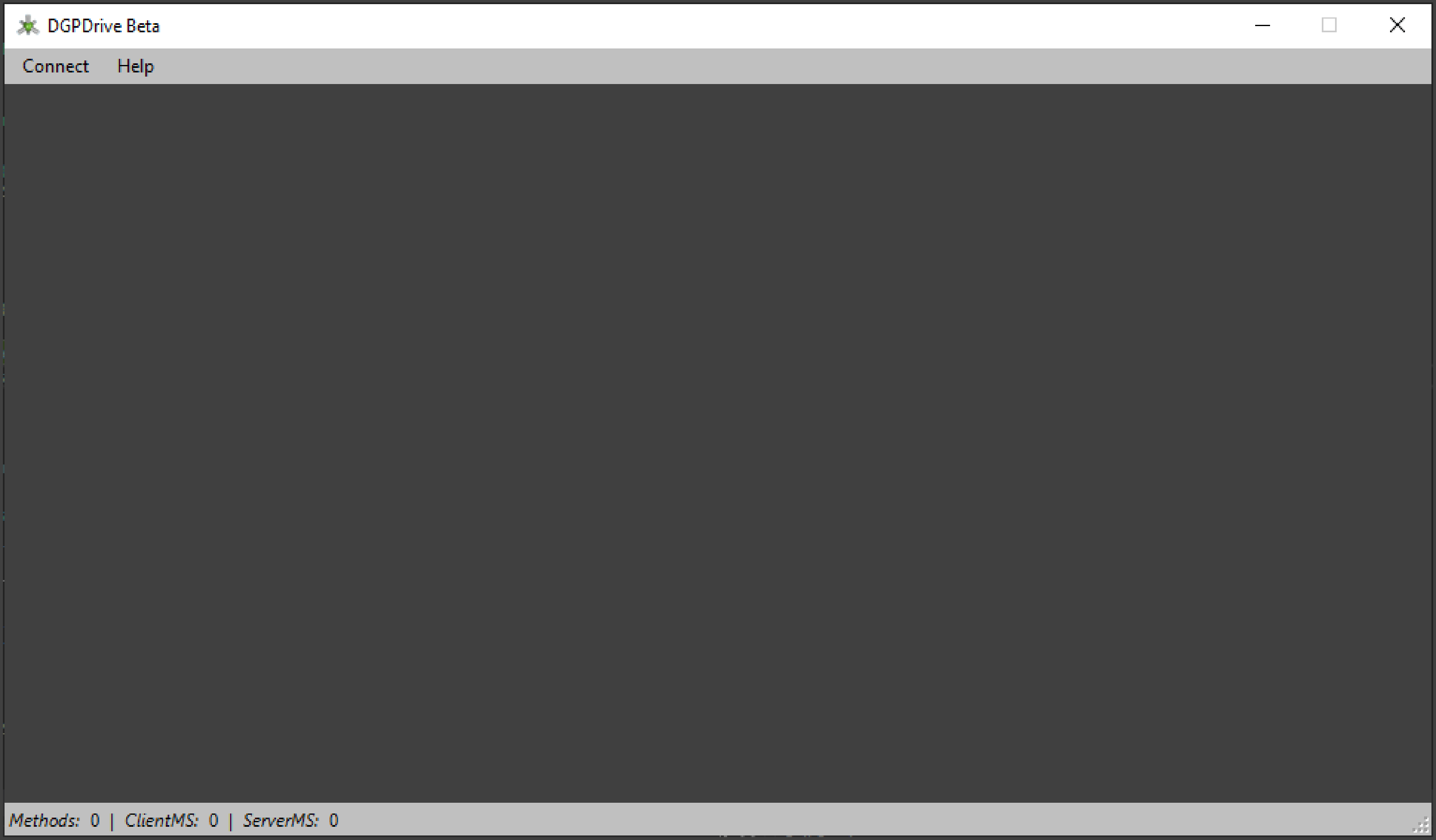
The DGPDrive application consists of all of the DGP admin UI's needed to manage the data-driven RBAC security, replication configuration, and test harness applications, plus the optional FileStore file sharing functionality. The DGPDrive app itself is basically a somewhat polished version of a fully functional prototype. A WinForm app was chosen for its Rapid Application Development capabilities in order to reduce application development timelines.

The main form of the DGPDrive application has a navigation menu at the top and a status bar at the bottom, with the majority of the UI in between devoted to the application workspace. All of the main sections of the application UI are implemented as WinForm user controls that fill the main workspace when called by the navigation menus. The user controls are very modular and largely self-contained. Some data is shared between the user controls via properties managed within the main form.

Navigation Menu

The navigation menu is customized to match the user's authorized roles when they connect to an environment of a DGP system. This has nothing to do with security, which is enforced within the web services. It is done to improve the user experience by not showing functionality that a user is not authorized to use when they are connected to a particular environment. Each environment has its own separate instances of the DGP databases, which will have different content from dev, to test, to QA, to prod. For example, a tester will be authorized to have access to more roles in the test environment than they would in the production environment of the same system, due to each environment having their own separate SysInfo security databases. The top navigation menu items shown will reflect those differences, and can be thought of as a high level version of feature toggles.

Virtually all of the functionality in the system is implemented as reusable web service API methods, with minimal functionality contained within the client application itself. This means that just about every action a user makes in the UI results in a call to a remote web service. Obviously, this type of architecture only works in practice if the web services are very fast.



Methods: 0 | ClientMS: 0 | ServerMS: 0

This screen cap shows the DGPDrive Main Form for the sysadmin account after using the Connect Form, which calls the Login Method. The Login method is a process that executes multiple steps on the server (which can often take 50 MS to complete even with caching).

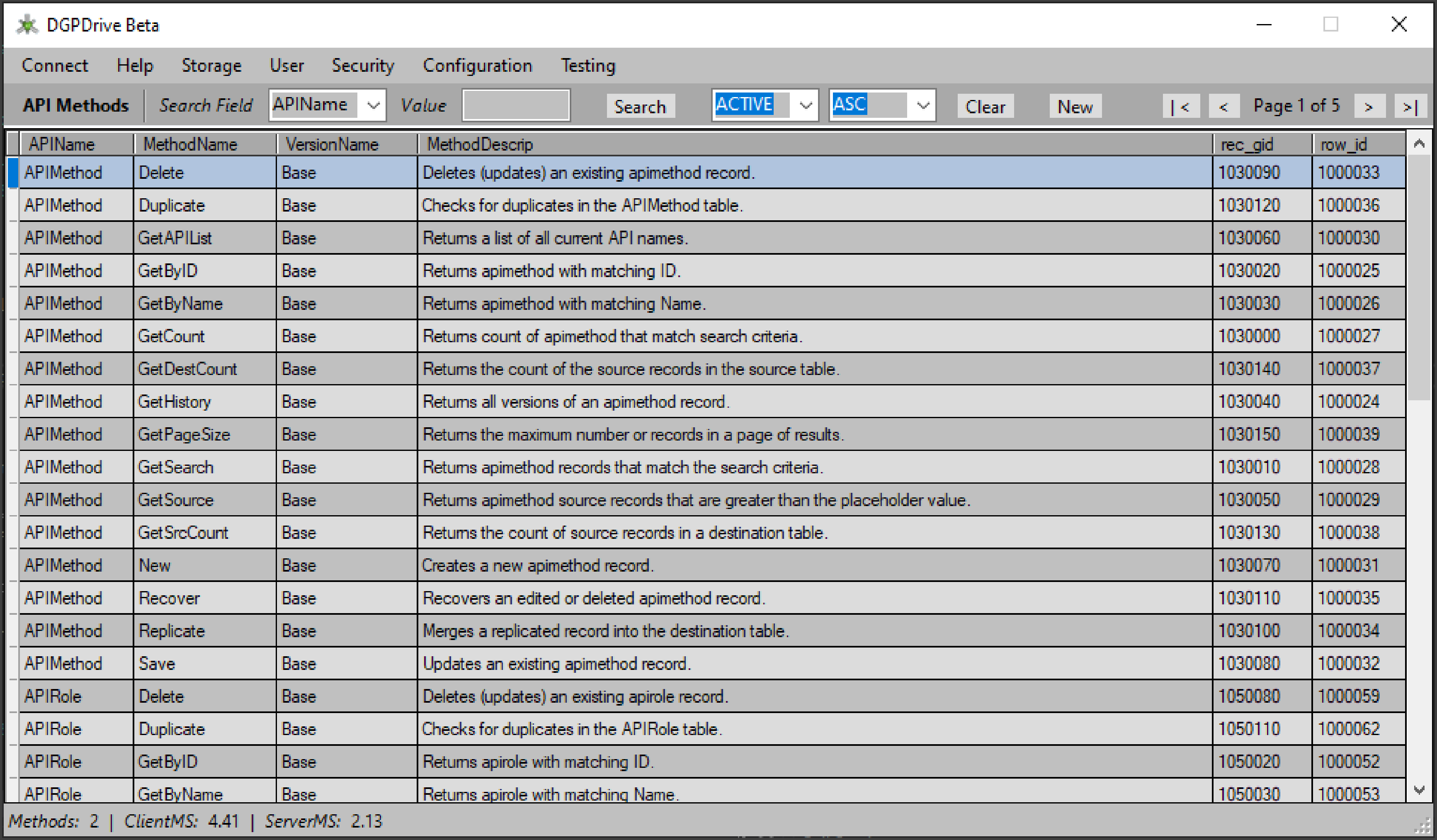
Status Bar

The purpose of the status bar at the bottom of the UI is to show the results of the end-to-end performance for the web service API calls made by each of the primary sections of the UI. It shows how many methods were called in the API request message batch, the total milliseconds of the end-to-end round trip from the client app to the server and back again, and the duration in milliseconds of the web service processing time for all internal methods executed on the server. The difference between the total end-to-end duration and the server-side processing time is roughly equivalent to the network latency for that API request.

Allowing users to constantly observe the performance of each environment of each system helps to emphasize the importance of high performance (and efficiency). The performance standards for all DGP systems is that every user interaction in an application should be completed in one second or less, which is explained very well in this video from a Google engineer responsible for mobile and web app performance: [https://www.youtube.com/watch?v=Il4swGfTOSM]. These performance standards are applicable to all types of applications, whether they are web or native.

In addition, poor performance is frequently one of the best indicators of “gray failures” in distributed systems. In those cases, some part of the distributed system is not functioning correctly, but the problem is not yet serious enough be throwing exceptions and only shows up as a slowdown in performance. The remote monitor functionality built into the DGPDrive application stores the status data to the system for a selected subset of users, which can then be analyzed and used to notify administrators when problems occur.

The effects of various levels of caching become evident when observing the performance numbers while using the application. The first time a system is called after starting up can take several seconds to respond, depending on how IIS and ASP.NET are configured, and whether or not the web services were precompiled when published, etc. Once that is taken into consideration, the JIT compilation of each method the first time it is called after a domain restart produces a noticeable performance difference compared to the second time the same method is called. In addition, the caching of the UserInfo object eliminates a query to the SysInfo database for each API request AND the need to decrypt the password, significantly improving performance. Finally, database caching also noticeably improves performance compared to the first time a query is executed by an API method. Refreshing a form to repeat an API method call will show performance increasing in steps as more levels of cache hits take effect.



APIName	MethodName	VersionName	MethodDescrip	rec_gid	row_id
APIMethod	Delete	Base	Deletes (updates) an existing apimethod record.	1030090	1000033
APIMethod	Duplicate	Base	Checks for duplicates in the APIMethod table.	1030120	1000036
APIMethod	GetAPIList	Base	Returns a list of all current API names.	1030060	1000030
APIMethod	GetByID	Base	Returns apimethod with matching ID.	1030020	1000025
APIMethod	GetByName	Base	Returns apimethod with matching Name.	1030030	1000026
APIMethod	GetCount	Base	Returns count of apimethod that match search criteria.	1030000	1000027
APIMethod	GetDestCount	Base	Returns the count of the source records in the source table.	1030140	1000037
APIMethod	GetHistory	Base	Returns all versions of an apimethod record.	1030040	1000024
APIMethod	GetPageSize	Base	Returns the maximum number or records in a page of results.	1030150	1000039
APIMethod	GetSearch	Base	Returns apimethod records that match the search criteria.	1030010	1000028
APIMethod	GetSource	Base	Returns apimethod source records that are greater than the placeholder value.	1030050	1000029
APIMethod	GetSrcCount	Base	Returns the count of source records in a destination table.	1030130	1000038
APIMethod	New	Base	Creates a new apimethod record.	1030070	1000031
APIMethod	Recover	Base	Recovers an edited or deleted apimethod record.	1030110	1000035
APIMethod	Replicate	Base	Merges a replicated record into the destination table.	1030100	1000034
APIMethod	Save	Base	Updates an existing apimethod record.	1030080	1000032
APIRole	Delete	Base	Deletes (updates) an existing apirole record.	1050080	1000059
APIRole	Duplicate	Base	Checks for duplicates in the APIRole table.	1050110	1000062
APIRole	GetByID	Base	Returns apirole with matching ID.	1050020	1000052
APIRole	GetByName	Base	Returns apirole with matching Name.	1050030	1000053

Methods: 2 | ClientMS: 4.41 | ServerMS: 2.13

As the same screens are called repeatedly, more and more levels of caching (and cache hits) take effect, resulting in a step-wise improvement in performance. The eventual best level of performance is shown in the screen cap above (localhost dev VM).

Remote Monitoring Data Collection

As was mentioned above, the main form of the DGPDrive UI has been designed to collect and store performance metrics while selected accounts use the DGPDrive application in real production systems. Collection and storage of these metrics is enabled for users that are members of the RemoteMonitor role. Care should be taken to insure that the number of accounts collecting performance metrics does not overwhelm the available storage or adversely affect the performance of the production system locations.

When enabled, the end-to-end performance metrics displayed for the major sections of the UI are also saved to the SysMetrics database, along with some other information about the computer and account used during each session.

DGPDrive App.config File

The default values for most of the entries in the DGPDrive app.config file do not require editing. The one exception to that rule could be the EventSource and EventID values. Those are needed in order to add entries to the Event Viewer logs if any problems occur in the standard logging mechanism (which is to store log entries in various database tables).

The default value "borrows" the use of an event source that already exists on all Windows computers and is rarely if ever seen in the Event Viewer, namely the .NET Runtime. A utility in the DGPDrive app can be used to create a new Event Viewer event source, if the user is able to run the app as an administrator on each computer that would need it. The app.config would then need to be edited to replace the ".NET Runtime" entries with the new event source and EventID values.

Field Name	Field Values	Description
Network	LOCALHOST, INTERNAL, DMZ, EXTERNAL, OFF	The network area where the AutoWork Test harness application is running
AutoWorkLogging	ON, OFF	A flag value to turn logging for AutoWork processes on or off
AutoWorkMaxDurMS	50	The max duration for the logic to claim and process queue records (used by the AutoWorkTester test harness)

EventSource	.NET Framework	The default event source to use for logging info to the Event Viewer if a custom event source has not been created
EventID	1000	The event ID to use when logging info to the default Event Viewer
LocFilePath		Default path to root folder containing work directories, test files, etc.
TestFilePath		The default path to the local folder storing test files