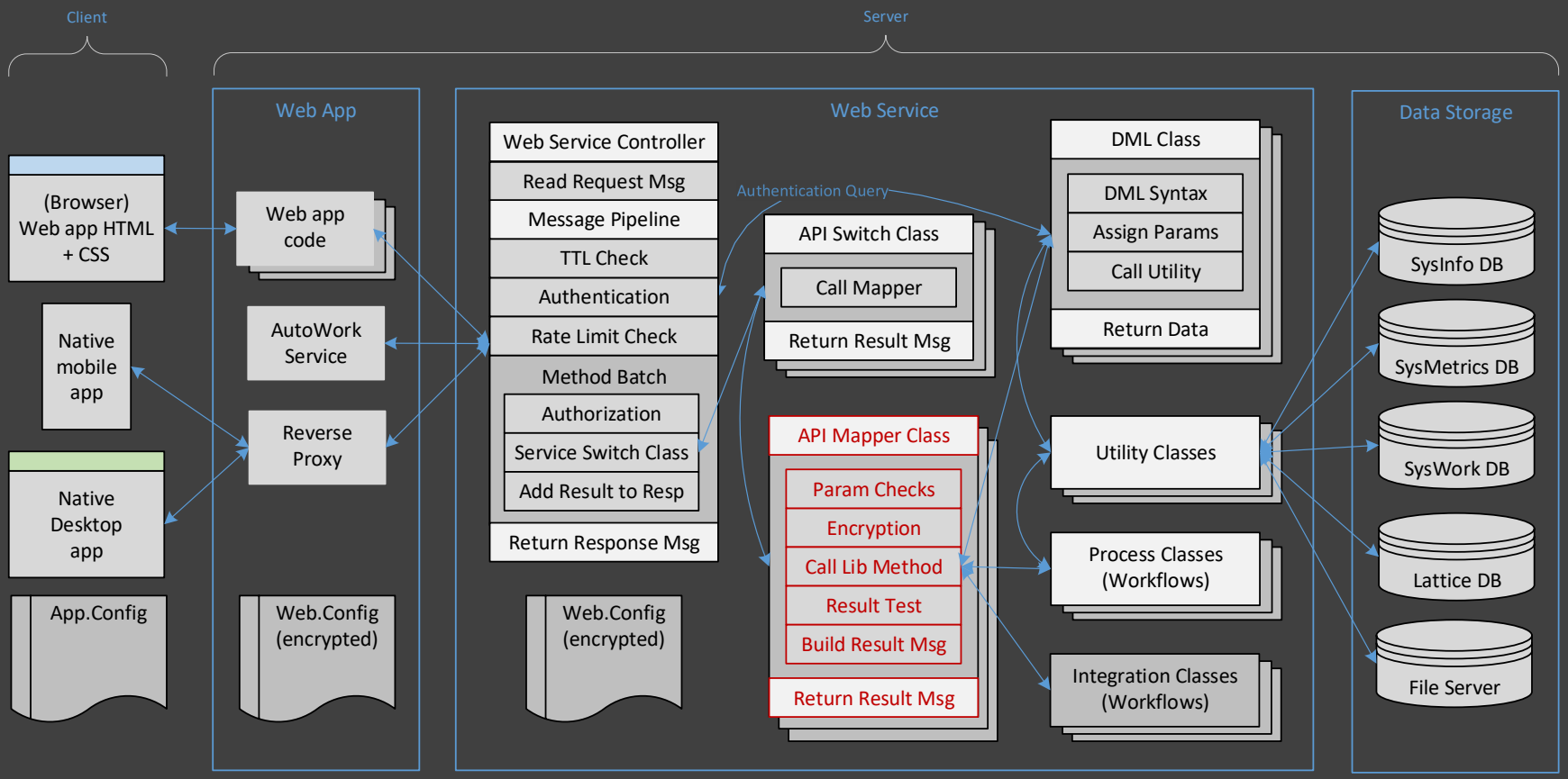

The API mapper classes act as the interface between the method calls of the API request message and calls to internal library methods that do all of the actual work in a system.

All mapper methods follow the same basic template for the following functionality:

1. Input Parameter Checks

The input parameters from the message are read into a generic dictionary of name/value pairs. The name of the input parameters are the key in the dictionary used to retrieve their corresponding values. The first block of functionality is to check for the existence of all required input parameters passed in by the API Request message, and optionally check their data type as well. Any missing or incorrect input parameters create an error message that is returned to the calling application. All input parameter values are text, and are cast to their correct data types as needed, when needed. The mapper method is also one of the places where optional parameters and default values for parameters can be managed.

2. Library Method Call

If the input parameter checks all pass, then one of the code library classes are instantiated. This call to the library method represents the layer of abstraction between the externally accessible API method names and the library methods that do all of the actual work in a system.

All of the message input parameters are mapped to the concrete method parameters of the library object. The convention is for no system logic to be contained in the mapper method itself, and the mapper method should call a single library method (no series of sequential steps in the mapper). In those situations, multiple steps are encapsulated into a process class method for better reusability by other internal libraries.

3. Evaluation of the Return Value

The value returned from the library method call is evaluated and mapped to one of the Result Codes (OK, Empty, Error, or Exception). The evaluation of the library method return value is a key feature that enables the end-to-end unit testing of the API Test harness. Testing the return value within the API method itself allows the logic to be centralized and implemented/maintained in one location in the code. The message-based API's in turn allow a two-way communication of much more data than a standard method call, so the evaluated result code can be returned along with the library method result value.

4. Build Result Message

The response message contains a collection of one or more result messages, each of which contains:

- Result name – each API method call can return multiple results, each of different types than the others. In the client application, each result is then retrieved by name from the collection of results in the response message.
- Result code – the result code from the evaluation of the library method return value.
- Data type – the data type of the result value
- Result Value – the value returned by the library method, or an error message if a problem occurred

Error Handling

Logic and data errors are handled in each mapper method. Error result messages are created and added to the collection of result messages.

Exceptions are also handled in each mapper method. Exception result messages are created and added to the collection of result messages, and the errors are logged using the ServerErrLog methods. These methods first write the exception to the Event Viewer, and then write the same exception to the centralized DGPError table.

Return the Result Message

The result message (positive or negative) for the library method call is returned to the message pipeline to be included in the response message.