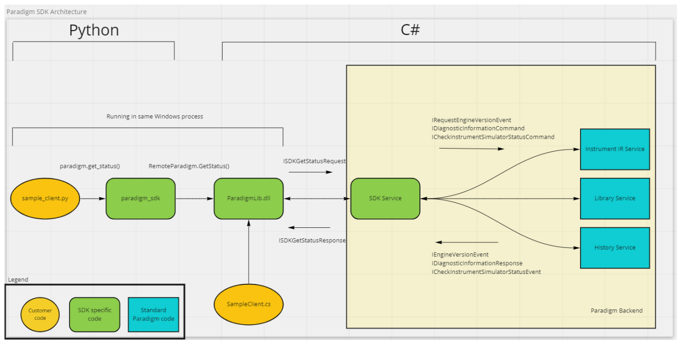
SDK Theory of Operation

Overview

This Software Development Kit (SDK) offers the means to control the OMNIC Paradigm backend programmatically. The available commands are described in detail in the “ParadigmLib” API documentation section.

The DLL files provided in this release can be thought of as a replacement for the OMNIC Paradigm client, where instead of a user communicating with the client by clicking buttons on a screen, a programmatic implementation is communicating with the backends directly via the provided commands. Programmatic control allows users to make custom (lightweight) applications that serve to control the instrument without having the OMNIC Paradigm client needlessly consuming resources on your machine.



Architecture

In the above Diagram 1, the customer application is shown to be written in Python. The custom sample_client.py file utilizes commands outlined in the Paradigm_SDK interface. This interface acts as a bridge between the language of development in which the client is written (in this case Python) and the ParadigmLib assembly which is compiled from C#. The ParadigmLib assembly will send the necessary instructions to the OMNIC Paradigm backends and handle any response that comes back, and subsequently forward that response on to the sample_client.py client. All of this interchange is abstracted from the user and can be thought of as simply controlling OMNIC Paradigm by sending the commands offered to you by the SDK interface.

Interactions With Traditional Client

Importantly, the traditional OMNIC Paradigm client is absent from the architecture in Diagram 1. Running the traditional client alongside your customized client may result in indeterminant behavior. Initiating a command from the Paradigm client that requires control of the instrument will disallow any instrument controlling commands from executing properly via your SDK client implementation.

Database Storage

Measurements initiated via SDK commands will have their results stored in the database the same way measurements initiated from the traditional OMNIC Paradigm client are. These measurements are stored and can be accessed for viewing or manipulation later. Measurements stored in the database can be configured with a retention policy, meaning that if this data is not intended to be utilized after a certain amount of time, it can be configured for deletion rather than permanently stored (needlessly taking up large amounts of storage on your machine).

Measurement Parameters

Currently, there is no way to customize parameters using the SDK, you must choose an existing named parameter set, or import and name a parameter set from a '.expx' file (these files can be easily generated by exporting settings from Paradigm to your file system). Otherwise, the traditional Paradigm Client must be used to customize and name your parameters.

Example Programs

We provide an example program for each supported language that can serve as a known-good starting point. Once the example code runs successfully, the developer can focus more on application logic and less on SDK configuration.

Each program starts by creating a RemoteParadigm object and initializing the connection to the OMNIC Paradigm backend. Once connected, there are a few different categories of functions that are available: calibration commands like AlignSpectrometer, data acquisition commands like StartSampleMeasurement, and processing commands like AdvancedATRCorrection. The program then closes the connection to the backend and exits.

While the SDK is simplest to run using C#, it is also possible to run with other languages such as Python. Python uses a package called pythonnet to be able to call C# (.NET) code from a regular Python program. The example for Python shows the necessary code to include and utilize this package as well. The files that contain example code are:

- C#.cs
- Python_client.py

Deployment

The process for deploying the Paradigm SDK along with your application is currently the same as setting up for development. In the future, there may be special steps for deployment and they will be included here. For example, we may develop a pared-down version of the SDK for deployment that takes up less space than a full-developer install.