About the framework and its purpose

The framework is meant to be a template that helps the user design processes that offer, at a barebones minimum, a way to store, read, and easily modify project configuration data, a robust exception handling scheme and event logging for all exceptions and relevant transaction information.

Because logs generated by each process are a vital component of its report generation, the framework logs messages at each relevant step toward solving a business transaction and sends those logs to the *Orchestrator server*. This is turn can be connected to the *ELK stack (Elasticsearch, logstash, kibana platform)* which enables data storage and countless ways of representing the data.

When we build tools, we try to first define their purpose and, in this scenario, the purpose of our framework is to solve a collection of business transactions. Notice i did not write business process, as all but the most simple business processes are typically composed of multiple, distinct in scope and in purpose, collections of business transactions. Thus, let us henceforth call such a collection of relatable business transactions a *business process component*, a part of a complete business process.

Thus, we could define a business process component as the sum of actions by which the data needed for a set of transactions is obtained, processed, and is input into or out of an *IT resource*.

Such a component needs to be easily deployed to the machines it will run on (Orchestrator server maintains versioning and easy deployment across all runtime machines), needs to be scalable and needs to be able to communicate it's output data with external mediums so that other components of the business process may pick up the work where it left off. Such a medium could be a shared folder, a data server, ftp server, email, Orchestrator server queue e.t.c.