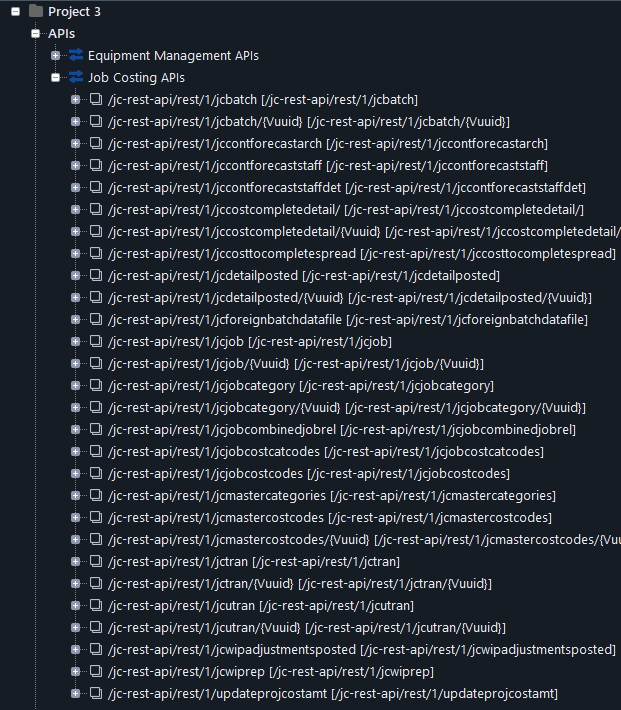
**API Standards Proof of Concept**

In our current state, if we want to create documentation, we must do it in 3 places to maintain the current process.

This would involve creating the manual documentation for ReadMe (cmicapi.com), creating a Postman collection, as well as adding the API and base tests to ReadyAPI so that QA has it for automated testing.

In an ideal world, we can update our APIs in a single place, and use that to automatically update the others. For our use case, this would be done via an OAS 3/Swagger specification file. This specification file is what is typically used to document REST/SOAP APIs and follows a standard – they are made in JSON or YAML formats but follow the same structure. Some already existing examples of this can be found in YAML format here: <https://github.com/Roshan-Sun/TestReadyAPIRepo/tree/main/YAML> or in JSON format here: <https://github.com/Roshan-Sun/TestReadyAPIRepo/tree/main/JSON>.

With an existing project, these specifications files can be imported into ReadyAPI quite easily and maintained with any updates that occur to the file. Take the below screenshot for example where the Job Costing APIs spec was imported:



You can see the format in which it is imported – all the APIs are formatted at the same level with no organization besides the separation by spec files. However, our currently existing project looks as follows:

A screenshot of a computer

Description automatically generated

You can see that they’re all held under the same webservices portion but sectioned off by application. However, this does not translate well into YAML/JSON since it doesn’t follow the expected format of OAS 3 or Swagger. This means implementing this idea would require restructuring of the current project that is being used by QA for automation testing. But once it has been restructured, it can be updated using the spec file as previously explained.

For ReadyAPI, these webservices are only 1 portion – there is a Functional Tests section where the actual test cases are written. These are what get run during the Jenkins automation or manual run to determine if webservices are working.

The spec files don’t account for this, so this will always require manual intervention on the developer’s behalf to get at least the base tests included for usage by QA. Please see screenshot below for reference:

A screenshot of a computer

Description automatically generated

Once all specifications files are included in a project, along with their respective Functional Tests, the entire ReadyAPI project can be used to maintain the project on top of the individual documentation files. In such a scenario, developers would follow the following steps:

1. Create any new documentation by modifying the spec file
2. Use the built-in functionality to update the webservices in ReadyAPI
3. Add any needed functional/baseline tests
4. Commit these changes to Git
5. Notify QA of the changes that have been made so their automated testing can be updated accordingly

The same specification file can also be imported into Postman – it will be accepted in either YAML or JSON format. Postman imports each specification file as its own collection and organizes it into its own format, breaking it down by the URL. In my opinion, it does a cleaner job of arranging the APIs in a folder format. You can see below the example of importing the same Job Costing specification as was done for ReadyAPI:

A screenshot of a computer

Description automatically generated

As soon you can see, a proper specification file can be maintained by all developers in a repository and be used to update and maintain their respective ReadyAPI projects and Postman collections.

Implementing this process will require quite a few changes considering our current process but will make it easier to document going forward and provide information to our end users regarding the APIs we have created.