Australian Government
Department of Climate Change, the Environment and Water

FullCAM Modernisation: Overview of the FullCAM External API's

What is an API?

An API (Application Programming Interface) is an interface that provides programmatic access to service functionality and data within an application or a database. In the context of APIs, Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses.

How do APIs work?

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. FullCAM implemented its API in REST Architecture.

What are REST APIs?

REST (Representational State Transfer) defines a set of functions like GET, PUT, POST, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP. The main feature of REST API is statelessness, whereby servers do not save client data between requests. Client requests to the server are like URLs you type in your browser to visit a website and the response from the server is plain data, without the typical graphical rendering of a web page.

FullCAM API

In order for external users to access the FullCAM application our team provides the FullCAM UI (User Interface) and FullCAM API. FullCAM UI is used to create plot files and run simulations and FullCAM API provides two set of APIs: a Data API which is used to get the spatial data, a Plot Simulation API to provision users to run the simulations using existing plot files.

Benefits of FullCAM API:

The Classic FullCAM application provides a downloadable .exe which allows users to run simulation only on windows OS (Operating System) and lacks the option to use the application on a non-windows platforms and web-based platforms. FullCAM2020PR APIs will provide the flexibility to use FullCAM in various web- and non-web based platforms. Users can utilize the API to gain Spatial data and run Simulations and can build applications which will remove the need to download and install an application to run simulations.

Audience

This guide will outline how to use the FullCAM API to gain access to spatial data and run simulations. APIs are created for the use of technical developers who are planning to use the API in their Frontend or desktop-based applications. It is assumed that the reader has knowledge of technical application development.

Access

You will need to send the FullCAM business team an email (fullcam@dcceew.gov.au) requesting access with the below information.

1. Org. Name:

2. Email:
3. Phone No.:
4. Business use case:
5. (Details to include No. of requests per system that are intended, No. of users, Peak time of usage. Etc.,)
6. No. of Users:
7. How long do you need access:

Once the request is issued, FullCAM will issue a Unique API subscription key that will be used to consume the API in your application.

How to use API:

API Documentation in the Developer Portal/Swagger Site provides a complete list of endpoints, and each endpoint will consist of the following:

- The parameters, both custom and standard supported (required and optional data which can be passed into the function call)
- What type of object the function will return. For example, the Service operation returns a 'Service Model' object.
- HTTP Status return codes supported (see section on HTTP Response Codes). Note that in the FullCAM implementation successful requests return a '200' status response code, even if no resources match the query (in which case an empty result set will be returned).

Best Practice Guide to Development

1. Ensure you define model objects (according to the Swagger documentation) to handle the appropriate response from the API call.
2. Configure the API endpoint base URLs. Send these Web Service requests and package the response into sets (such as lists) of the relevant model objects returned.
3. To avoid a poor user experience of large delays, call the Web Services asynchronously rather than synchronously.
4. Ensure you know when the results from Web Service requests are OK and when there is a problem (and the nature of the problem) by defining mechanisms to handle the HTTP status codes results – e.g. detecting and responding to code 404, which means the requested endpoint wasn't found.
5. To improve search performance, cache all relevant data such as run simulation locally, so you can search it quickly rather than have to do a request to the server for each request for information. Refresh your cache daily to get the latest data.