Overview

Previous Up Next

This section gives an overview of the steps involved in using ecFlow.

Step 1: Write a suite definition

The *suite definition* describes how your tasks run and interact. *task*s can be grouped together in families, which themselves may be placed in other families and/or *suite*s. All the entities (tasks, families and suites) are called *node*s and form a hierarchical tree.

There are two main methods for describing a *suite definition* to the *ecflow_server*.

- via a **text** *suite definition*

  The grammar of this text definition is described by *Definition file Grammar*. This grammar does not support conditional statements (such as if, while, for) nor the ability to define functions. However, the text definition file can be generated/created using any language which in itself supports conditional statements. The text definition is similar to that offered by SMS/CDP and as such may be an appropriate migration path for some users.
- via a **Python** *suite definition*

  This allows more checking and functionality and as such is our **preferred** method. See *ecFlow Python Api*.

Step 2: Write your task scripts

*ecf script*s are text files that correspond to the *task* in the *suite definition*. The script defines the **main work** that is to be carried out. The script includes *child command*s, special comments, and manual sections that provide information for users.

The *child command*s are a restricted set of *ecflow_client* commands that communicate with the *ecflow_server*. They inform the server when the job has started, completed, aborted, or set some *event*.

Step 3: Start an ecFlow server

After *ecflow_server* is started, the *suite definition* can then be loaded into it.

- The user then initiates *scheduling* in the *ecflow_server*
- *scheduling* will check *dependencies* in the *suite definition* every minute (by default). If these *dependencies* are free, the server will submit the *task*. This process is called *job creation*. The running process corresponding to the *task* is referred to as a job.

The running jobs will communicate back to the server using *child command*s. These cause:

- *status* changes on the *node*s held in the server.
- update to attributes of a node (i.e. like *event*s, *meter*s, and *label*s)

Step 4: Interact with the GUI

ecFlow has a specialised GUI client, called *ecflow_ui* This is used to visualise and monitor:

- The hierarchical structure of the *suite definition*. (*suite*, *family*, *task*)
- State changes in the nodes and servers.
- Attributes of the nodes and any *dependencies*.
- *ecf script* file and the expanded *job file*.

In addition, *ecflow_ui* provides a rich set of *ecflow_client* commands that can interact with the server.

⚠ The following tutorial will show examples in plain text and Python. However, it is **recommended** that you use Python, since the later tutorial examples use conditionals like ‘if’ and looping constructs.

Previous Up Next