20 Steps

The core of the RAVEN calculation flow is the **Step** system. The **Step** is in charge of assembling different entities in RAVEN (e.g. Samplers, Models, Databases, etc.) in order to perform a task defined by the kind of step being used. A sequence of different **Steps** represents the calculation flow.

Before analyzing each **Step** type, it is worth to 1) explain how a general **Step** entity is organized, and 2) introduce the concept of step "role" . In the following example, a general example of a **Step** is shown below:

```
<Simulation>
  ...
  <Steps>
    ...
    <WhatEverStepType name='aName'>
        <Role1 class='aMainClassType'
           type='aSubType'>userDefinedName1</Role1>
        <Role2 class='aMainClassType'
           type='aSubType'>userDefinedName2</Role2>
        <Role3 class='aMainClassType'
           type='aSubType'>userDefinedName3</Role3>
        <Role4 class='aMainClassType'
           type='aSubType'>userDefinedName4</Role4>
    </WhatEverStepType>
    ...
  </Steps>
  ...
</Simulation>
```

As shown above each **Step** consists of a list of entities organized into "Roles." Each role represents a behavior the entity (object) will assume during the evaluation of the **Step**. In RAVEN, several different roles are available:

- **Input** represents the input of the **Step**. The allowable input objects depend on the type of **Model** in the **Step**.
- **Output** defines where to collect the results of an action performed by the **Model**. It is generally one of the following types: **DataObjects**, **Databases**, or **OutStreams**.
- **Model** represents a physical or mathematical system or behavior. The object used in this role defines the allowable types of **Inputs** and **Outputs** usable in this step.

- **Sampler** defines the sampling strategy to be used to probe the model.
  It is worth to mention that, when a sampling strategy is employed, the "variables" defined in the `<variable>` blocks are going to be directly placed in the **Output** objects of type **DataObjects** and **Databases**).
- **Function** is an extremely important role. It introduces the capability to perform pre or post processing of Model **Inputs** and **Outputs**. Its specific behavior depends on the **Step** is using it.
- **ROM** defines an acceleration Reduced Order Model to use for a **Step**.
- **SolutionExport** represents the container of the eventual output of a step. For the moment, there are two uses: 1) A **Step** is employing the search of the Limit Surface (LS), through the class of Adaptive **Samplers**); in this case, it contains the coordinates of the LS in the input space; and 2) Some of the post-processors employ clustering algorithms and the cluster centers will be stored in this file with the input being the cluster labels.

Depending on the **Step** type, different combinations of these roles can be used. For this reason, it is important to analyze each **Step** type in details.

The available steps are the following

- SingleRun (see Section 20.1)
- MultiRun(see Section 20.2)
- IOStep(see Section 20.3)
- RomTrainer(see Section 20.4)
- PostProcess(see Section 20.5)

20.1 SingleRun

The **SingleRun** is the simplest step the user can use to assemble a calculation flow: perform a single action of a **Model**. For example, it can be used to run a single job (Code Model) and collect the outcome(s) in a "**DataObjects**" object of type **Point** or **History** (see Section 14 for more details on available data representations).

The specifications of this Step must be defined within a `<SingleRun>` XML block. This XML node has the following definable attributes:

- `name`, ***required string attribute***, user-defined name of this **Step**. **Note:** This name is used to reference this specific entity in the `<RunInfo>` block, under the `<Sequence>` node. If

the name of this **Step** is not listed in the **<Sequence>** block, its action is not going to be performed.

- **pauseAtEnd**, ***optional boolean/string attribute (case insensitive)***, if True (True values = True, yes, y, t), the code will pause at the end of the step, waiting for a user signal to continue. This is used in case one or more of the **Outputs** are of type **OutStreams**. For example, it can be used when an **OutStreams** of type **Plot** is output to the screen. Thus, allowing the user to interact with the **Plot** (e.g. rotate the figure, change the scale, etc.).
  Default: False.

In the **<SingleRun>** input block, the user needs to specify the objects needed for the different allowable roles. This step accepts the following roles:

- **<Input>**, ***string, required parameter***, names an entity (defined elsewhere in the RAVEN input) that will be used as input for the model specified in this step. This XML node accepts the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. For example, **'Files'**, **'DataObjects'**, **'Databases'**, etc.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the main object class. For example, if the **class** attribute is **'DataObjects'**, the **type** attribute might be **'PointSet'**. **Note:** The class **'Files'** has no type (i.e. **type=''**).

  Note: The **class** and, consequently, the **type** usable for this role depends on the particular **<Model>** being used. In addition, the user can specify as many **<Input>** nodes as needed.
- **<Model>**, ***string, required parameter***, names an entity defined elsewhere in the input file to be used as a model for this step. This XML node accepts the following attributes:
  - **class**, ***required string attribute***, main object class type. For this role, only **'Models'** can be used.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the `Models` object class. For example, the **type** attribute might be **'Code'**, **'ROM'**, etc.
- **<Output>**, ***string, required parameter*** names an entity defined elsewhere in the input to use as the output for the **Model**. This XML node recognizes the following attributes:
  - **class**, ***required string attribute***, main object class type. For this role, only **'DataObjects'**, **'Databases'**, and **'OutStreams'** can be used.

- `type`, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the main object class. For example, if the `class` attribute is `'DataObjects'`, the `type` attribute might be `'PointSet'`.

Note: The number of `<Output>` nodes is unlimited.

Example:

```
<Steps>
  ...
  <SingleRun name='StepName' pauseAtEnd='false'>
        <Input    class='Files'
          type=''>anInputFile.i</Input>
        <Input    class='Files'          type=''>aFile</Input>
        <Model   class='Models'      type='Code'>aCode</Model>
        <Output  class='Databases' type='HDF5'>aDatabase</Output>
        <Output  class='DataObjects'
          type='History'>aData</Output>
  </SingleRun>
  ...
</Steps>
```

20.2 MultiRun

The **MultiRun** step allows the user to assemble the calculation flow of an analysis that requires multiple "runs" of the same model. This step is used, for example, when the input (space) of the model needs to be perturbed by a particular sampling strategy.

The specifications of this type of step must be defined within a `<MultiRun>` XML block. This XML node recognizes the following list of attributes:

- `name`, ***required string attribute***, user-defined name of this Step. **Note:** As with other objects, this name is used to reference this specific entity in the `<RunInfo>` block, under the `<Sequence>` node. If the name of this **Step** is not listed in the `<Sequence>` block, its action is not going to be performed.
- `pauseAtEnd`, ***optional boolean/string attribute***, if True (True values = True, yes, y, t), the code will pause at the end of the step, waiting for a user signal to continue. This is used in case one or more of the **Outputs** are of type **OutStreams**. For example, it can be used when an **OutStreams** of type **Plot** is output to the screen. Thus, allowing the user to interact with the **Plot** (e.g. rotate the figure, change the scale, etc.).
- `sleepTime`, ***optional float attribute***, in this attribute the user can specify the waiting time (seconds) between two subsequent inquiries of the status of the submitted job (i.e. check if

a run has finished).
Default: 0.05.

In the **<MultiRun>** input block, the user needs to specify the objects that need to be used for the different allowable roles. This step accepts the following roles:

- **<Input>**, ***string, required parameter***, names an entity to be used as input for the model specified in this step. This XML node accepts the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. For example, **'Files'**, **'DataObjects'**, **'Databases'**, etc.
  - **type**, ***required string attribute***, the actual entity type. This attribute specifies the object type within the main object class. For example, if the **class** attribute is **'DataObjects'**, the **type** attribute might be **'PointSet'**. **Note:** The class **'Files'** has no type (i.e. **type=''**).

  Note: The **class** and, consequently, the **type** usable for this role depend on the particular **<Model>** being used. The user can specify as many **<Input>** nodes as needed.
- **<Model>**, ***string, required parameter*** names an entity defined elsewhere in the input that will be used as the model for this step. This XML node recognizes the following attributes:
  - **class**, ***required string attribute***, main object class type. For this role, only **'Models'** can be used.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the `Models` object class. For example, the **type** attribute might be **'Code'**, **'ROM'**, etc.
- **<Sampler>**, ***string, optional parameter*** names an entity defined elsewhere in the input file to be used as a sampler. As mentioned in Section 12, the **Sampler** is in charge of defining the strategy to characterize the input space. This XML node recognizes the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used. Only **'Samplers'** can be used for this role.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the `Samplers` object class. For example, the **type** attribute might be **'MonteCarlo'**, **'Adaptive'**, **'AdaptiveDET'**, etc. See Section 12 for all the different types currently supported.
- **<Optimizer>**, ***string, optional parameter*** names an entity defined elsewhere in the input file to be used as an optimizer. As mentioned in Section 13, the **Optimizer** is in charge of defining the strategy to optimize an user-specified variable. This XML node recognizes the following attributes:

- **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used. Only **'Optimizers'** can be used for this role.
- **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the Optimizers object class. For example, the **type** attribute might be **'SPSA'**, etc. See Section 13 for all the different types currently supported.

Note: For Multi-Run, either one **<Sampler>** or one **<Optimizer>** is required.

- **<SolutionExport>**, ***string, optional parameter*** identifies an entity to be used for exporting key information coming from the **Sampler** or **Optimizer** object during the simulation. This XML node accepts the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. For this role, only **'DataObjects'** can be used.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the DataObjects object class. For example, the **type** attribute might be **'PointSet'**, **'HistorySet'**, etc.
    Note: Whether or not it is possible to export the **Sampler** solution depends on the **type**. Currently, only the Samplers in the **'Adaptive'** category and all Optimizers can export their solution into a **<SolutionExport>** entity. For Samplers, the **<Outputs>** node in the DataObjects needs to contain the goal **<Function>** name. For example, if **<Sampler>** is of type **'Adaptive'**, the **<SolutionExport>** needs to be of type **'PointSet'** and it will contain the coordinates, in the input space, that belong to the "Limit Surface". For Optimizers, the **<SolutionExport>** needs to be of type **'HistorySet'** and it will contains all the optimization trajectories, each as a history, that record how the variables are updated along each optimization trajectory.

- **<Output>**, ***string, required parameter*** identifies an entity to be used as output for this step. This XML node recognizes the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. For this role, only **'DataObjects'**, **'Databases'**, and **'OutStreams'** may be used.
  - **type**, ***required string attribute***, the actual entity type. This attribute specifies the object type within the main object class. For example, if the **class** attribute is **'DataObjects'**, the **type** attribute might be **'PointSet'**.

Note: The number of **<Output>** nodes is unlimited.

Example:

```
<Steps>
  ...
  <MultiRun name='StepName1' pauseAtEnd='False' sleepTime='0.01'>
    <Input class='Files' type=''>anInputFile.i</Input>
    <Input class='Files' type=''>aFile</Input>
    <Sampler class='Samplers' type = 'Grid'>aGridName</Sampler>
    <Model class='Models' type='Code'>aCode</Model>
    <Output class='Databases' type='HDF5'>aDatabase</Output>
    <Output class='DataObjects' type='History'>aData</Output>
  </MultiRun >
  <MultiRun name='StepName2' pauseAtEnd='True' sleepTime='0.02'>
    <Input   class='Files' type=''>anInputFile.i</Input>
    <Input   class='Files' type=''>aFile</Input>
    <Sampler class='Samplers' type='Adaptive'>anAS</Sampler>
    <Model   class='Models' type='Code'>aCode</Model>
    <Output  class='Databases' type='HDF5'>aDatabase</Output>
    <Output  class='DataObjects' type='History'>aData</Output>
    <SolutionExport class='DataObjects' type='PointSet'>
      aTPS
    </SolutionExport>
  </MultiRun>
  ...
</Steps>
```

20.3 IOStep

As the name suggests, the **IOStep** is the step where the user can perform input/output operations among the different I/O entities available in RAVEN. This step type is used to:

- construct/update a *Database* from a *DataObjects* object, and vice versa;
- construct/update a *DataObject* from a *CSV* file contained in a directory;
- construct/update a *Database* or a *DataObjects* object from *CSV* files contained in a directory;
- stream the content of a *Database* or a *DataObjects* out through an **OutStream** object (see section 16);
- store/retrieve a *ROM* to/from an external *File* using Pickle module of Python.

This last function can be used to create and store mathematical model of fast solution trained to predict a response of interest of a physical system. This model can be recovered in other simulations or used to evaluate the response of a physical system in a Python program by the implementing of the Pickle module. The specifications of this type of step must be defined within an **<IOStep>** XML block. This XML node can accept the following attributes:

- **name**, ***required string attribute***, user-defined name of this Step. **Note:** As for the other objects, this is the name that can be used to refer to this specific entity in the **<RunInfo>** block, under the **<Sequence>** node.
- **pauseAtEnd**, ***optional boolean/string attribute (case insensitive)***, if True (True values = True, yes, y, t), the code will pause at the end of the step, waiting for a user signal to continue. This is used in case one or more of the **Outputs** are of type **OutStreams**. For example, it can be used when an **OutStreams** of type **Plot** is output to the screen. Thus, allowing the user to interact with the **Plot** (e.g. rotate the figure, change the scale, etc.).
  Default: False.
- **fromDirectory**, ***optional string attribute***, The directory where the input files can be found when loading data from a file or series of files directly into a DataObject.

In the **<IOStep>** input block, the user specifies the objects that need to be used for the different allowable roles. This step accepts the following roles:

- **<Input>**, ***string, required parameter***, names an entity that is going to be used as a source (input) from which the information needs to be extracted. This XML node recognizes the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. As already mentioned, the allowable main classes are **'DataObjects'**, **'Databases'**, **'Models'** and **'Files'**.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the main object class. For example, if the **class** attribute is **'DataObjects'**, the **type** attribute might be **'PointSet'**. If the **class** attribute is **'Models'**, the **type** attribute must be **'ROM'** and if the **class** attribute is **'Files'**, the **type** attribute must be **' '**
- **<Output>**, ***string, required parameter*** names an entity to be used as the target (output) where the information extracted in the input will be stored. This XML node needs to contain the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. The allowable main classes are **'DataObjects'**, **'Databases'**, **'OutStreams'**, **'Models'** and **'Files'**.
  - **type**, ***required string attribute***, the actual entity type. This attribute specifies the object type within the main object class. For example, if the **class** attribute is **'OutStreams'**, the **type** attribute might be **'Plot'**.

This step acts as a "transfer network" among the different RAVEN storing (or streaming) objects. The number of **<Input>** and **<Output>** nodes is unlimited, but should match. This step assumes a 1-to-1 mapping (e.g. first **<Input>** is going to be used for the first **<Output>**, etc.).
Note: This 1-to-1 mapping is not present when **<Output>** nodes are of **class 'OutStreams'**, since **OutStreams** objects are already linked to a Data object in the relative RAVEN input block.

In this case, the user needs to provide in the **<Input>** nodes, all of the "DataObjects" objects linked to the OutStreams objects (see the example below):

```
<Steps>
  ...
  <IOStep name='OutStreamStep'>
    <Input class='DataObjects'
       type='HistorySet'>aHistorySet</Input>
    <Input class='DataObjects' type='PointSet'>aTPS</Input>
    <Output class='OutStreams' type='Plot'>plot_hist
    </Output>
    <Output class='OutStreams' type='Print'>print_hist
    </Output>
    <Output class='OutStreams' type='Print'>print_tps
    </Output>
    <Output class='OutStreams' type='Print'>print_tp
    </Output>
  </IOStep>
  <IOStep name='PushDataObjectsIntoDatabase'>
    <Input  class='DataObjects'
       type='HistorySet'>aHistorySet</Input>
    <Input  class='DataObjects'     type='PointSet'>aTPS</Input>
    <Output class='Databases' type='HDF5'>aDatabase</Output>
    <Output class='Databases' type='HDF5'>aDatabase</Output>
  </IOStep>
  <IOStep name='ConstructDataObjectsFromCSV'>
    <Input class='Files' type=''>aCSVFile</Input>
    <Output class='DataObjects' type='PointSet'>aPS</Output>
  </IOStep>
  <IOStep name='ConstructDataObjectsFromDatabase'>
    <Input class='Databases' type='HDF5'>aDatabase</Input>
    <Input class='Databases' type='HDF5'>aDatabase</Input>
    <Output class='DataObjects'
       type='HistorySet'>aHistorySet</Output>
    <Output class='DataObjects'    type='PointSet'>aTPS</Output>
  </IOStep>
```

```
  <IOStep name='PushROMIntoFile'>
    <Input class='Models'  type='ROM'>aROM</Input>
    <Output class='Files'  type=''>aFile</Output>
  </IOStep>
  <IOStep name='ImportROMFromFile'>
    <Input class='Files'  type=''>zFile</Input>
    <Output class='Models'  type='ROM'>zROM</Output>
  </IOStep>
  ...
</Steps>
```

Example of how to use a ROM exported by RAVEN using the IOStep, where (x1,x2) are the input variable names for which the ROM has been trained: Example Python Function:

```
import os
from glob import glob
import inspect
import utils
import numpy as np
for dirr,_,_ in os.walk("path_to_framework"):
 utils.add_path(dirr)
import pickle
fileobj = open('zFile','rb')
unpickledObj = pickle.load(fileobj)
dictionary={"x1":np.atleast_1d(Value1),"x2":np.atleast_1d(Value2)}
eval=unpickledObj.evaluate(dictionary)
print str(eval)
fileobj.close()
```

20.4 RomTrainer

The **RomTrainer** step type performs the training of a Reduced Order Model. The specifications of this step must be defined within a **<RomTrainer>** block. This XML node accepts the attributes:

- **name**, ***required string attribute***, user-defined name of this step. **Note:** As for the other objects, this is the name that can be used to refer to this specific entity in the **<RunInfo>** block under **<Sequence>**.

In the **<RomTrainer>** input block, the user will specify the objects needed for the different allowable roles. This step accepts the following roles:

- **<Input>**, ***string, required parameter*** names an entity to be used as a source (input) from which the information needs to be extracted. This XML node accepts the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. The only allowable main class is **'DataObjects'**.
  - **type**, ***required string attribute***, the actual entity type. This attribute specifies the object type within the main object class. For example, the **type** attribute might be **'PointSet'**. **Note:** Since the ROMs currently present in RAVEN are not time-dependent, the only allowable types are **'Point'** and **'PointSet'**.
- **<Output>**, ***string, required parameter***, names a ROM entity that is going to be trained. This XML node recognizes the following attributes:
  - **class**, ***required string attribute***, main object class type. This string corresponds to the tag of the main objects type used in the input. The only allowable main class is **'Models'**.
  - **type**, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the main object class. The only type accepted here is, currently, **'ROM'**.

Example:

```
<Steps>
  ...
  <RomTrainer name='aStepName'>
    <Input  class='DataObjects'  type='PointSet'>aTPS</Input>
    <Output class='Models' type='ROM'          >aROM</Output>
  </RomTrainer>
  ...
</Steps>
```

20.5 PostProcess

The **PostProcess** step is used to post-process data or manipulate RAVEN entities. It is aimed at performing a single action that is employed by a **Model** of type **PostProcessor**.

The specifications of this type of step is defined within a **<PostProcess>** XML block. This XML node specifies the following attributes:

- `name`, ***required string attribute***, user-defined name of this Step. **Note:** As for the other objects, this is the name that is used to refer to this specific entity in the `<RunInfo>` block under the `<Sequence>` node.
- `pauseAtEnd`, ***optional boolean/string attribute (case insensitive)***, if True (True values = True, yes, y, t), the code will pause at the end of the step, waiting for a user signal to continue. This is used in case one or more of the **Outputs** are of type **OutStreams**. For example, it can be used when an **OutStreams** of type **Plot** is output to the screen. Thus, allowing the user to interact with the **Plot** (e.g. rotate the figure, change the scale, etc.).
  Default: False.

In the `<PostProcess>` input block, the user needs to specify the objects needed for the different allowable roles. This step accepts the following roles:

- `<Input>`, ***string, required parameter***, names an entity to be used as input for the model specified in this step. This XML node accepts the following attributes:
  - `class`, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. For example, `'Files'`, `'DataObjects'`, `'Databases'`, etc.
  - `type`, ***required string attribute***, the actual entity type. This attribute specifies the object type within the main object class. For example, if the `class` attribute is `'DataObjects'`, the `type` attribute might be `'PointSet'`. **Note:** The class `'Files'` has no type (i.e. `type=''`).

  Note: The `class` and, consequently, the `type` usable for this role depends on the particular type of **PostProcessor** being used. In addition, the user can specify as many `<Input>` nodes as needed by the model.
- `<Model>`, ***string, required parameter***, names an entity to be used as a model for this step. This XML node recognizes the following attributes:
  - `class`, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input. For this role, only `'Models'` can be used.
  - `type`, ***required string attribute***, the actual entity type. This attribute needs to specify the object type within the `'Models'` object class. The only type accepted here is `'PostProcessor'`.
- `<Output>`, ***string, required/optional parameter***, names an entity to be used as output for the PostProcessor. The necessity of this XML block and the types of entities that can be used as output depend on the type of **PostProcessor** that has been used as a **Model** (see section 17.5). This XML node specifies the following attributes:
  - `class`, ***required string attribute***, main object class type. This string corresponds to the tag of the main object's type used in the input.

– **type**, ***required string attribute***, the actual entity type. This attribute specifies the object type within the main object class. For example, if the **class** attribute is **'DataObjects'**, the **type** attribute might be **'PointSet'**.

Note: The number of **<Output>** nodes is unlimited.

Example:

```
<Steps>
  ...
  <PostProcess name='PP1'>
      <Input  class='DataObjects'  type='PointSet' >aData</Input>
      <Model  class='Models' type='PostProcessor'>aPP</Model>
      <Output class='Files'  type=''>anOutputFile</Output>
  </PostProcess>
  ...
</Steps>
```