TN on applicability and guidelines for the design of interfaces compliant to the "*ESA generic E2E simulator Interface Control Document*" (PE-ID-ESA-GS-464)

Michele Zundo EOP-PEP
v1.0

The "*ESA generic E2E simulator Interface Control Document*" [E2EGICD] allows extensive freedom in how the data structure for an openSF compliant interface can be defined to support standardized orchestration of the sw modules; in addition, there are choices that should be considered to maximise flexibility, interface extension, use of standard tools/framework and end-user usability. This TN collects the lessons learned from past developments.

The most important is the **explicit exposure of inputs (files) and parameters** at the level of the generic interface (as opposed to being hidden inside lower-level files, referred by path or inside data structures, that is required to achieve this objective and specify a well-defined data interface.

Selected sections, relevant for the module interface understanding, are extracted from the [E2EGICD] here below (in red) and commented to highlight the critical points.

Please ensure you have read and understood the [E2EGICD] before reading below.

The interfaces described in [E2EGICD] are shown below in Figure 2-1.

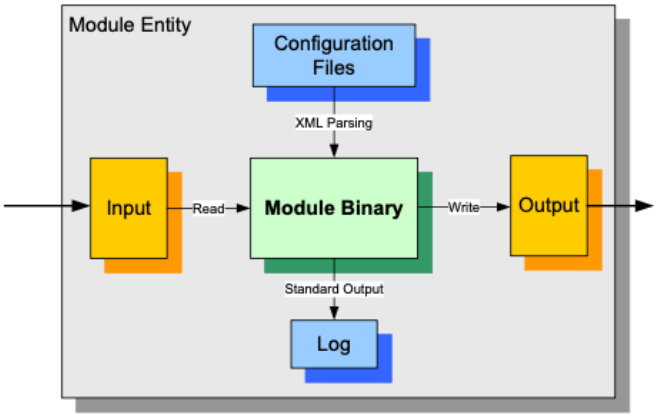

Figure 2-1: Module Entity Interfaces

1. ***Input/Output Files (section 2.2.1)*** ***Figure 2-1 orange box***

There are no constraints imposed on modules for reading input files or writing output files. Modules have the freedom to specify as input/output either a directory or a set of one or more files. In case a directory name is used, it is responsibility of each module to select the relevant/correct inputs from the directory. A compliant orchestration infrastructure (e.g. openSF) will automatically trigger a module execution when all the input files are detected as present, and the generating process has completed successfully.

Recommendation:
Preference to use files, instead of directories, as input in the command line unless the files cannot be specified statically

Examples:
a) Module_1 generate an output file called `my_product.nc` which is then used by Module_2 as input.

Recommendation:
Define statically the input to Module 2 as '`my product.nc`'

b) Module 1 generates as output a variable number of files (e.g. depending on data inside some of his inputs) called: `my_product_01.nc, my_product_02.nc, my_product_03.nc,`

Recommendation:
Define a directory called `my_product/` and define both output of Module_1 and input of Module 2 the path `my_product/`. Module_1 will then write its out file(s) into the directory and Module_2 will be invoked and only given the path and line unless the files cannot be specified statically

To note that no check/verification at all is done by the orchestrating framework on how many files or if there are files at all inside the directory, but only that the directory is created/exists and the responsibility lies with the module to detect any error condition.

2. ***Auxiliary Files (section 2.2.2)***

A compliant E2E orchestrating framework makes use of one separate directory per each simulation execution where it will copy the required input files not generated by the modules (e.g. input for the first module or initial data) . This copy includes directories and their content whenever the input is specified as such. To avoid disk space overuse the developer can define large input files as "auxiliary files".
An "auxiliary file" is therefore a data file that is used as input to the simulation/processing chain but that will not be duplicated and stored in the simulation execution folder e.g. of candidates are a very large orbit files, Databases, LUTs or reference data files which do not change (e.g. physical constants). The "auxiliary files" are consequently not declared in the command line as inputs but passed to the module as a parameter within the configuration file described in section 2.2.6) and therefore not processed/visible by the orchestrating framework.

NB Declaring input data as auxiliary files should be carefully assessed as it makes the data set within the execution directory incomplete and does not ensure that re-execution will be identical, as external auxiliary data could have been changed and reduce the robustness of the E2E simulation chain as the existence of auxiliary files is not ensured by the orchestrating framework.

Recommendation:
Define small/normal or changing Auxiliary files as normal inputs (Figure 2-1 orange box) and instead bypass the copying of Auxiliary data that are both big and static (e.g. Digital Elevation Model) by referring to its path inside the Configuration Files (Figure 2-1 blue box).

3. ***XML configuration files (section 2.2.6)*** **Figure 2-1 blue box**

The behaviour of a given module can be controlled using the two ***XML configuration files*** *as previously described. This section describes the rules and conventions that such configuration files have to follow in addition to standard [XML] rules, e.g. the preamble <?xml version=”1.0” encoding=”UTF-8”?>.*

a) Format of XML [E2EGICD] Configuration Files

A generic library called OSFI to read/write parameter files in the correct format as well as a convenient editor (called Parameter Editor PE) is provided as part of the openSF package at eop-cfi.esa.int.

b) What is a Configuration Parameter?

Parameters within the Configuration files are *CONFIGURATION* parameters to control/modify the execution of SW Modules in the E2ES and not meant to be input data.
Attention should be made to identify and clearly separate parameters for execution of simulation, sensitivity analysis, processing and what-if scenario run (e.g. noise to be added, biases, algorithm selection, thresholds, logging or processing behavior) from the input data needed for the processing.

Do not use the parameters in the Configuration files to pass auxiliary data, orbit data, instrument characterisation, etc.. Use input files for this purpose.

c) Location of parameters

The decision of how many and which are the parameters included in the Configuration files compliant to [E2EGICD] is critical since **only parameters included there are visible externally to the orchestrating framework and to supporting tools.**

Having parameters in any other place e.g. in bespoke files passed to modules as input (**orange box**) or as string/path within the [E2EGICD] defined configuration file (**blue box)** makes them invisible and useless to the orchestration and to a framework user. This prevents running of automatic sensitivity analysis, configuration management of the changed parameters and use of existing tool to directly modify/revert/save parameters configuration in different versions/runs.

Recommendation:
Put all the needed configuration parameters inside the XML Configuration files as per [E2EICD] and have the modules read the parameters from the XML configuration files using OSFI.

d) Parameter identification

The list of parameters that will be defined as configuration parameters depends on the modelling but should be extensive enough to give flexibility to modify execution behaviour and be iterated with the E2ES users and developer to ensure all possible cases are covered (e.g. it might not make sense from the perspective of module1 to change the noise figure of a simulated element however this might be useful for developer of module2 that uses module1 as input.)

e) Parameter file structure

For readability and maintenance parameters should be grouped inside the Configuration files for each module per domain e.g. parameters related to platform should be a separate group from parameters related to environment from parameters related to instrument, etc.

An example of structure from a current mission:

```
<?xml version="1.0" encoding="UTF-8"?>
<MYE2ES_ISM_Local_Configuration version="01.00.00">
    <General>
        <parameter description="Instrument configuration data directory" name="InstrumentDataDirectory" type="STRING">./data</parameter>
        <!-- Below boolean vector entries are mapped to the following log level types: (ERROR, WARNING, INFO, DEBUG, PROGRESS) -->
        <parameter description="Log level flags" name="LogLevelFlags" dims="5" type="BOOLEAN">TRUE TRUE TRUE TRUE TRUE</parameter>
        <parameter description="Selection of Parallelised Simulation Mode" name="Parallel_Simulation_Mode" type="BOOLEAN">FALSE</parameter>
    </General>
    <Instr1_Parameters>
        <parameter description="Apply interferometric axis variation to the science beam" name="ApplyIAVariationForScienceBeam" type="BOOLEAN">FALSE</parameter>
        <parameter description="Apply interferometric axis variation to the reference beam" name="ApplyIAVariationForReferenceBeam" type="BOOLEAN">FALSE</parameter>
        <parameter description="Apply random speed fluctuations" name="ApplyRandomSpeedFluctuations" type="BOOLEAN">TRUE</parameter>
        <parameter description="Apply dynamic shear effects" name="ApplyDynamicShearEffects" type="BOOLEAN">TRUE</parameter>
        <parameter description="Apply effect of laser line width" name="ApplyLaserLineWidthEffect" type="BOOLEAN">TRUE</parameter>
        <parameter description="Use polaration distinction" name="UsePolarizationDistinction" type="BOOLEAN">TRUE</parameter>
        <parameter description="Apply electronic noise to the science signals" name="ApplyElectronicNoiseScienceSignal" type="BOOLEAN">FALSE</parameter>
        <parameter description="Apply electronic noise to the reference signals" name="ApplyElectronicNoiseReferenceSignal" type="BOOLEAN">FALSE</parameter>
        <parameter description="Split output into separate interferograms (one for each dwell)" name="SplitInterferogram" type="BOOLEAN">TRUE</parameter>
        <parameter description="BTDA sampling rate [Hz]" name="BTDASamplingRate" type="INTEGER">6510</parameter>
        <parameter description="Random case for the IA random speed variations" name="IaSpeedRandomCase" type="INTEGER">2</parameter>
        <parameter description="Random case for the IA random dynamic shear variations" name="IaDynamicShearRandomCase" type="INTEGER">2</parameter>
        <parameter description="Random case for the laser line width effect" name="LaserLineRandomCase" type="INTEGER">6</parameter>
        <parameter description="Random case for the noise on the electronic science 1 signal" name="Science1ElectronicRandomNoiseCase" type="INTEGER">3</parameter>
        <parameter description="Random case for the noise on the electronic science 2 signal" name="Science2ElectronicRandomNoiseCase" type="INTEGER">3</parameter>
        <parameter description="Random case for the noise on the electronic reference 1 signal" name="Reference1ElectronicRandomNoiseCase" type="INTEGER">3</parameter>
        <parameter description="Random case for the noise on the electronic reference 2 signal" name="Reference2ElectronicRandomNoiseCase" type="INTEGER">3</parameter>
        <parameter description="The UTC end time of the simulation [s] (use 282.0 to reduce to one dwell)" name="GlobalTimeEnd" type="FLOAT">450.80</parameter>
    </Instr1_Parameters>
    <Platform_Parameters>
        <parameter description="Set the platform attitude noise" name="Platform_Attitude_noise" type="FLOAT">12e-3</parameter>
        <parameter description="Set the platform OBC drift [microsecond/day]" name="Platform_OBC_drift" type="FLOAT">0</parameter>
    </Platform_Parameters>
</MYE2ES_ISM_Local_Configuration>
```