ATTEST

Ref. Ares(2023)7151227 - 20/10/2023

WP6

ICT Platform to enhance TSO/DSO coordination

Integration of the open-source toolbox

D6.2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 864298.

DOCUMENT CONTROL PAGE

DOCUMENT	D6.2 Integration of the open-source toolbox
TYPE	Other
DISTRIBUTION LEVEL	Public
DUE DELIVERY DATE	31 / 12 / 2022
DATE OF DELIVERY	31 / 01 / 2023
VERSION	V1.0
DELIVERABLE RESPONSIBLE	SOFTLAB
AUTHOR(S)	Clara Bagnasco, Christian Biasuzzi, Davide Longo, Massimo Ferraro, Lara Pinato
OFFICIAL REVIEWER(S)	Eduardo Martinez-Cesena, Hrvoje Keko

DOCUMENT HISTORY

VERSION	AUTHORS	DATE	CHANGES
0.1	Christian Biasuzzi, Davide Longo, Massimo Ferraro, Lara Pinato	15/10/2022	First version of D6.2: index
0.2	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	30/11/2022	Introduction
0.3	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	21/12/2022	Tools wrappers
0.4	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	03/01/2023	Fixes
0.5	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	13/01/2023	Installation guide,
0.6	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	17/01/2023	Fixes Services list
0.7	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	18/01/2023	Version to be reviewed
0.8	Eduardo Martinez-Cesena, Hrvoje Keko	24/01/2023	Internal review
1.0	Christian Biasuzzi, Massimo Ferraro, Lara Pinato	24/01/2023	Final version

Table of Contents

Abbreviations and Acronyms

API	Application Programming Interface
CLI	Command Line Interface
CSV	Comma-Separated Values
DB	Database
DSO	Distribution System Operators
GUI	Graphical User Interface
ICT	Information and Communication Technologies
JSON	JavaScript Object Notation
ODS	Open Document Sheet
PNG	Portable Network Graphics
REST	Representational State Transfer
TRL	Technology Readiness Level
TSO	Transmission System Operators

1. Introduction

The Advanced Tools Towards cost-efficient decarbonisation of future reliable Energy SysTems (ATTEST) project developed a modular open-source toolbox: a suite of tools to support Transmission System Operators (TSOs) and Distribution Systems Operators (DSOs) operate, maintain, and plan the energy systems. A key novelty of the tools is the consideration of flexibility services traded at the TSO/DSO interfaces.

The open-source toolbox has been embedded into an ICT platform that provides data access connectors and converters, tools orchestration functionalities and visualization interfaces, as shown in the picture below.

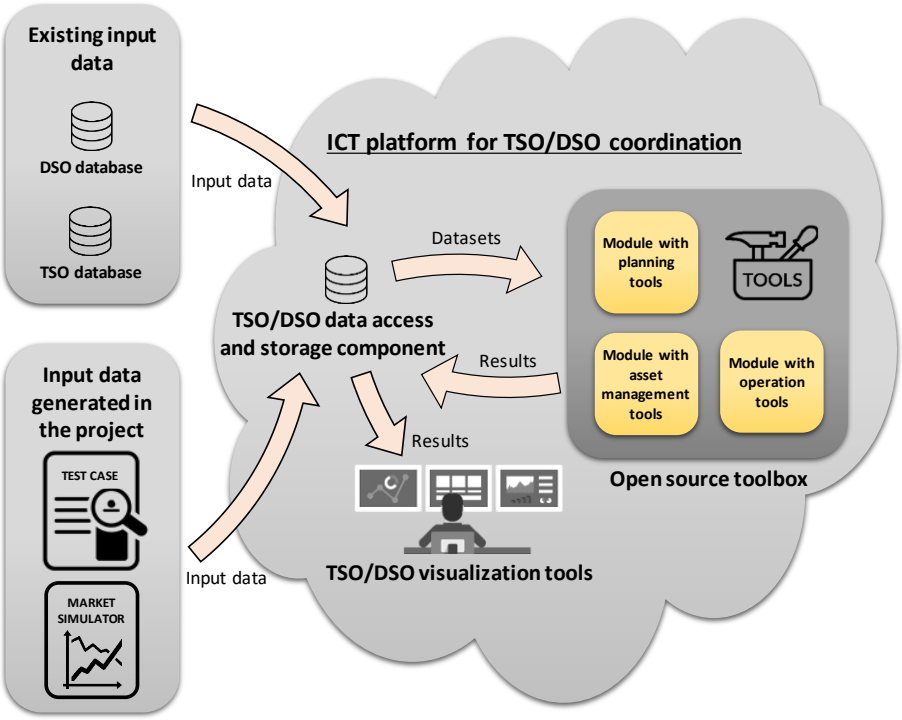

FIGURE 1 - ICT PLATFORM

Inside the ICT platform, the integration component allows the execution of all the tools from the open-source toolbox. For each tool, this component provides a wrapper that, at first, calls the ICT data access component to prepare the inputs required. Afterwards, the component runs the tool and finally gets its outputs and stores them into the ICT database.

This document accompanies the release of the software implementing the integration component, providing information about it. The rest of the document is structured as follows:

- Section 2 outlines the general approach that was taken for the integration of the different ATTEST tools, e.g., controller, java based, Conda multiplatform multiple environments.
- Section 3 details the tools wrappers and the changes to the code required to make the integration feasible.
- Section 4 describes the services that allow to run the tools.

- Section 5 provides information about the technologies used for the implementation of the component.
- The appendices contain the installation guide for the tools in the open-source toolbox.

A bird's-eye view on the toolbox and its requirements can be found in D2.2 Toolbox Specifications.

The tools are detailed in these documents:

- D3.2 Optimization tool for distribution network planning
- D3.3 Optimization tool for transmission network planning
- D3.4 Optimization tool for planning TSO/DSO shared technologies
- D4.2 Tool for ancillary services procurement in day-ahead operation planning of the distribution network
- D4.3 Tool for ancillary services activation in real-time operation of the distribution network
- D4.4 Tool for state estimation of distribution networks
- D4.5 Tool for ancillary services procurement in day-ahead operation planning of the transmission network
- D4.6 Tool for ancillary services activation in real-time operation of the transmission network
- D4.7 Tool for on-line dynamic security assessment
- D5.2 Tool for the characterization of the condition of assets
- D5.3 Tool for the definition of condition indicators based on heterogeneous information sources.
- D5.4 Tool for the definition of smart asset management strategies.

Besides, each document contains a user guide that explains how to install and run the respective tool, as a standalone software.

2. Integrating the tools of the open-source toolbox

Each tool in the open-source toolbox is mirrored in the ICT platform by a wrapper that drives the interactions with the tool. The tools wrappers are then used, for example, to run the tools from the ICT GUI.

To simplify the integration within the ICT Platform, the approach chosen throughout the project has been to install all ATTEST tools in the same, properly sized, machine. More complex approaches could be selected, for example, to accommodate heavier computation scenarios, by executing a subset of the tools in dedicated server. There is no conceptual restriction, the wrappers can well reside in different machines: in such case, the wrappers developed in this project should be extended according to the deploying choices for the tools.

The tools in the ATTEST's open-source toolbox have been developed during the project lifespan using multiple technologies: Python[1], Julia[2] , supported by some specialized optimization software packages like IBM's CPLEX[3], AMPL[4], and Ipopt[5].

It was chosen not to enforce a specific version of a software to let the tools developers concentrate on their task and not worry about integration issues. The tools are developed (for the most part) in an academic

[1] https://www.python.org/
[2] https://julialang.org/
[3] https://www.ibm.com/it-it/analytics/cplex-optimizer
[4] https://ampl.com/ce/
[5] https://coin-or.github.io/Ipopt/

setting so while our approach may seem unwieldy at first, it actually ensures that the progress in lower TRLs is more significant. As a result, different tools could use different versions of the same technology, e.g., different versions of the Python interpreter and of the libraries used. To avoid the conflicts between the tools it was selected an environment management: Conda[6], and specifically Miniforge[7], a lighter installer for the package manager. With Conda, each tool from the open-source toolbox works in a separate, dedicated, environment, that includes all the required dependencies needed for its execution. In fact, using each tool in its specific virtual environment is the recommended development approach, especially in Python development, in order to avoid dependency issues.

With this approach, it was possible to have the tools installed on the same machine, all available to be used by the ICT platform.

For each tool, its corresponding ICT Platform wrapper is in charge of collecting and preparing the input files from the ICT Platform data component (which includes data stored in the ICT DB and data provided by the users via the ICT GUI), executing the tool from the command line, collecting the outputs from the output files and storing the results back to the data component. The input and output file formats depend on the tool and include excel, ODS, CSV, JSON, PNG.

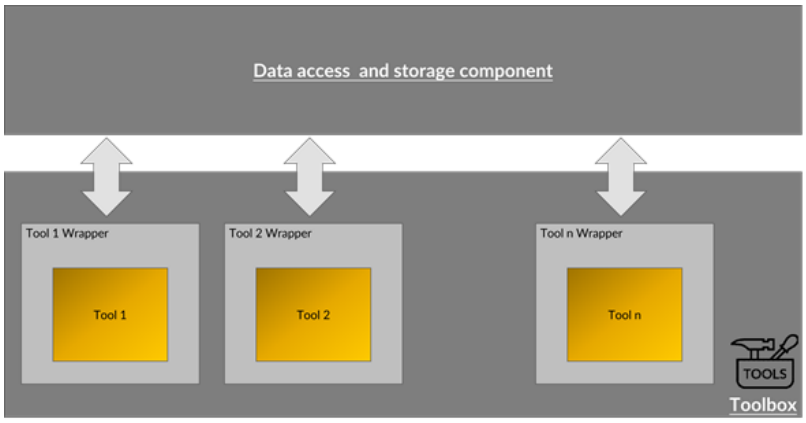

FIGURE 2 - TOOLS WRAPPERS

The ICT GUI uses the ICT integration component's services to interface the open-source toolbox's tools.

3. Tools wrappers details

Each tool in the open-source toolbox is assumed to be executed from the CLI (Command Line Interface). Input data and configurations can be provided to the tool as files and/or command line parameters. When executed, a tool computes its results and writes its outputs to one or more files.

The integration of the toolbox in the context of T6.2 required some changes to the tools code. Usually, the changes were aimed at simplifying the tool's wrapper: to allow the specification of parameters using the JSON format through a command line script, or to make the tool read and write data from specific directories, more suited to the ICT Platform's needs. In some cases, the changes were implemented to make the tool compliant with the model described above: e.g., replacing an interaction with a user used to drive the execution with an additional command line parameter to the tool.

[6] https://docs.conda.io/en/latest/
[7] https://github.com/conda-forge/miniforge

An installer script is provided for each tool. The installer takes care of creating the Conda environment for the tool and installing the required dependencies. It is important to note that some of the tools rely on commercial software (e.g., IBM CPLEX). These dependencies are not covered by the installer script, therefore they must be installed according to the specific tool instructions. The automated installation is not possible as the licensing of these tools requires manual installation.

A launcher script (i.e., launch.bat) is also provided for each tool; its purpose is to activate the Conda dedicated environment for the tool, before invoking its execution. When a tool wrapper is invoked with a set of input data parameters, it:

1. creates a new working directory, in a dedicated area of the filesystem.
2. creates the input data files structure, inside the working directory, as expected by the tool. The input data come from the ICT data component and the inputs provided by the users through the GUI.
3. Spawns an asynchronous process that executes the tool using its dedicated launcher script and waits for its completion.
4. Retrieves the results from the expected output files in the working directory.
5. Saves the outputs in the ICT data component.

Notes

- each run of a tool reads and writes to a dedicated working directory. Since the working directory's name is guaranteed to have a unique name, the approach allows multiple, concurrent, executions of the same tool, without conflicts. The console output for the tool execution is also redirected to a file that can be used for debugging purposes.

4. ICT tools related services

The ICT platform provides a set of services that can be used to execute the tools wrappers. These services are used by the ICT platform GUI to interface the open-source toolbox.

The services are implemented using REST WEB services technology and provided as a set of RESTful APIs. Details of the service architecture can be found in deliverable D6.1 "TSO/DSO data access and storage component".

Services are classified in three main categories:

- Prepare the tool inputs from the service parameters and run the tool wrapper, asynchronously.
- Retrieve the execution state of the tools.
- Retrieve the execution results.

The ICT tools services are protected through authentication and secure protocols.

More details on the APIs of the most significant services can be found in the Appendix ICT integration services.

5. Technologies

The ICT platform's integration layer, like other platform's components, has been developed using Java and related technologies. A detailed list of the technologies used can be found in D6.1 "TSO/DSO data access and storage component".

As mentioned above, Conda has been used as the software to be able to run the tools in separate environments.

6. Appendices

6.1. Installation guide

The installation guide of the ICT platform integration component's java part is already reported in the appendices of D6.1 "TSO/DSO data access and storage component". This section is meant to integrate that guide for what concerns the installation of the tools from the open-source toolbox and the Conda environment management software.

As already mentioned in D6.1, the installation guide targets a MS WINDOWS server, since the tools in the opensource toolbox were developed in WINDOWS OS

6.1.1. Requirements

Please note that the Installation of some of these components requires Administrator's privileges.

6.1.1.1. Conda

The Miniforge Conda installer for WINDOWS can be downloaded here:

https://github.com/conda-forge/miniforge

6.1.1.2. Julia

Some of the tools require Julia. Version 1.6.7 (LTS) for Windows can be downloaded here:

https://julialang-s3.julialang.org/bin/winnt/x64/1.6/julia-1.6.7-win64.zip

To minimize the changes to the configurations, please unzip the archive in c:\ATTEST

6.1.1.3. Third-party commercial software

Installation of third-party software with commercial licenses used by some of the tools (e.g., IBM CPLEX) is not covered in this guide. Please refer to the tool's user manuals and the specific software's instructions.

6.1.1.4. Open-source toolbox

For installation convenience, the tools with the integration related changes can be downloaded here:

https://github.com/ATTEST-project/attest-ict-tools

Each directory in the git repository contains a tool from the open-source toolbox. An installer.bat script for each tool is also provided, to help creating the Conda environment and installing the required libraries. The Conda software is assumed to be already installed (see Section 6.1.1.1).

The installation target is the directory C:\ATTEST\tools. Please make sure that the user that is installing the tools has the right privileges to write and execute files in that location and its subfolders.

6.2. ICT platform integration services

The ICT platform integration services are aimed at interfacing the tools from the open-source toolbox. For example, these services are used for running a tool, and downloading the results of the computation. This section details the main services exposed by the ICT platform integration component.

An extensive list of these services, with the complete REST APIs documentation, is provided by the ICT platform GUI, at the address: “localhost:9000/admin/docs”, in section:

tool-wp-< WorkPackageNum >-resource

Please note that a user must be authenticated with the ICT platform, to access the documentation.

6.2.1.Tools execution

The tools-execution related services are organized into work packages. The common end-point pattern is:

POST /api/tools/wp<WorkPackageNum>/run

Therefore, the endpoints for the run services are:

POST /api/tools/wp2/run
POST /api/tools/wp3/run
POST /api/tools/wp4/run
POST /api/tools/wp5/run

Usually, the common work-package service can dispatch the request to the correct tool, using the tool’s name passed as an input parameter.

In some cases, however, the tool’s name is included directly in the service endpoint. For example:

POST /api/tools/wp3/t32/run
POST /api/tools/wp3/t33/run

Of course, the specific tool’s input data and other configurations must be provided to the service as additional POST parameters.

6.2.2.Tools results

The tool results service returns a .zip archive containing all files generated by the tool, for a given task id.

GET /api/tasks/tool-results/:id

Where Id is a unique task identifier

6.2.3.Tools display results

The tools display results services return data in a JSON file, used to show a chart representation of the results generated by a tool, and it is used by the ICT Platform visualization components.

GET /api/tools/wp<WorkPackageNum>/show-charts

The request parameters used by each service, depend on the specific tool’s needs.

6.2.4.Tools execution log

The Tools execution log service returns the logfile generated by the tool, for a given task id.

GET api/tool-log-files/taskId/:id

Where Id is a unique task identifier

6.2.5.Tasks list

When run using the APIs listed in 6.2.1, the corresponding service creates an asynchronous task that maintain the tool's execution status. The tasks list service returns a list of tasks and their status information.

GET /api/tasks