



IMPACT MONITOR

CPACS tool integration

Tutorial

M. Alder | DLR



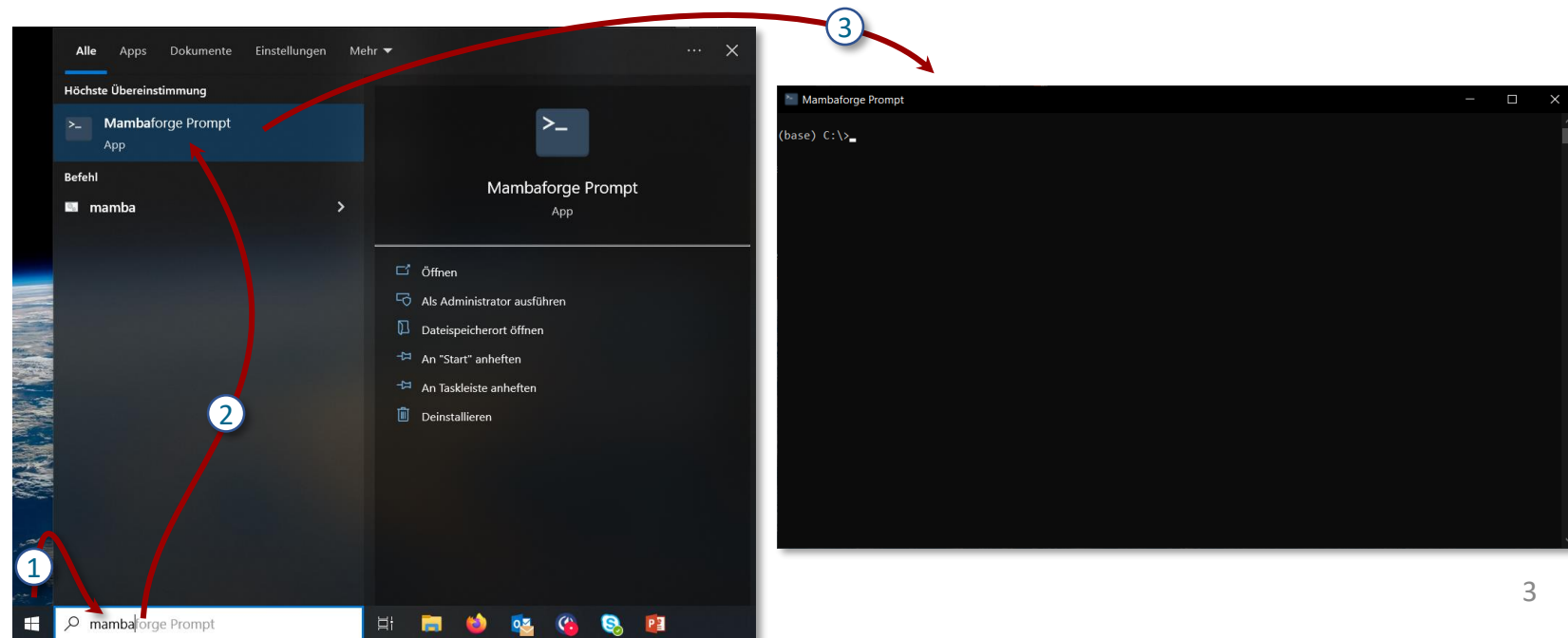
- **Aim of the exercise:** Adapting **your tool** to be able to:
 - Open a CPACS file
 - Add a log entry to CPACS
 - Write a CPACS file
- **Note:** Each tool has its own individual software architecture, which we want to influence as little as possible. The following steps are an example and may be different for you. The goal of writing a valid XML file (CPACS) should be the same for everyone.

Set-up your Python environment (1)

We recommend using [TiXI](#) for reading and writing CPACS files (available for C, C++, Fortran, JAVA and Python). Alternative XML libraries should be available for every major programming language.

The following steps can be skipped if you have already attended the [CPACS seminar](#).

1. Check if you have [Conda](#) installed. If not, you can install it via [Miniconda](#) or [Miniforge](#).



Set-up your Python environment (2)

- Create a Python environment (=a specific combination of Python version plus additional modules) with TiXI 3:

```
Mambaforge Prompt
(base) C:\>conda create -n cpacsEnv python=3.10 tixi3 -c dlr-sc
```

This will add TiXI 3 from DLR-SC

Install Python 3.10, but also other versions are supported

Choose your preferred environment name

```
conda-forge/noarch::ucrt
conda-forge/win-64::vc
conda-forge/win-64::vc14_runtime
conda-forge/win-64::vs2015_runtime
conda-forge/noarch::wheel
conda-forge/win-64::xz
Proceed ([y]/n)? y
```

Confirm with „y“

```
#
# To deactivate an active environment
#
# $ conda deactivate

(base) C:\>conda activate cpacsEnv
```

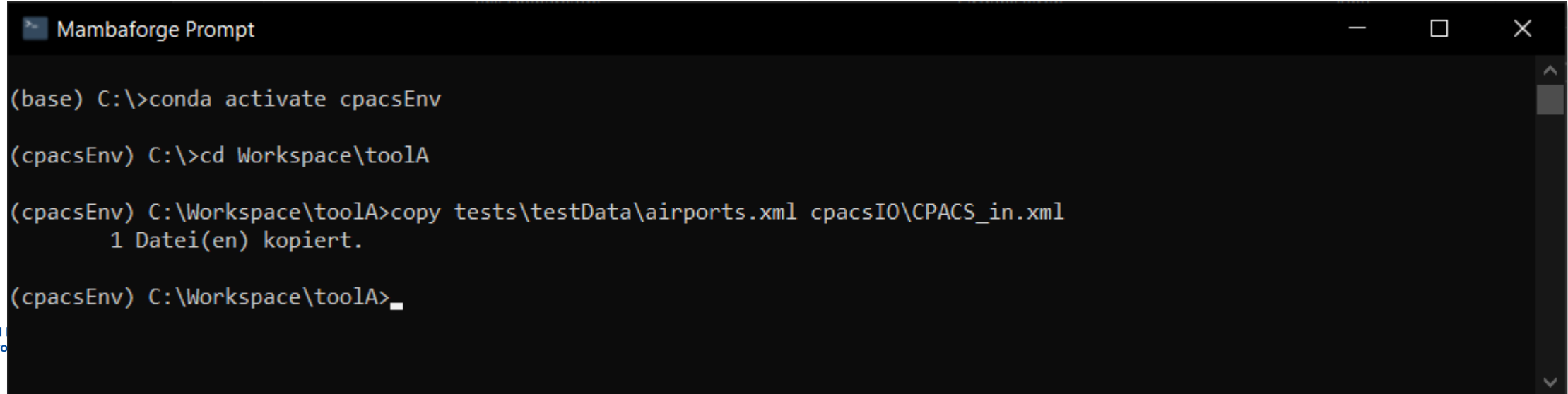
You are still in the „base“ environment. Don't forget to activate „cpacsEnv“

```
(cpacsEnv) C:\>python
Python 3.10.12 | packaged by conda-forge | (main, Jun 23 2023, 22:34:57)
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

(cpacsEnv) C:\>python
```

Test your Python environment (1)

1. Extract *example_tool.zip* (The tool is just called *toolA*, as we will later add exercises with additional tools, like *toolB* ;-)).
2. Navigate to this directory
3. Copy the sample CPACS file from *./testData/airports.xml* to *./cpacsIO/CPACS_in.xml* (later, RCE will do this step automatically for your tool).

A screenshot of a terminal window titled "Mambaforge Prompt". The terminal shows the following commands and output:

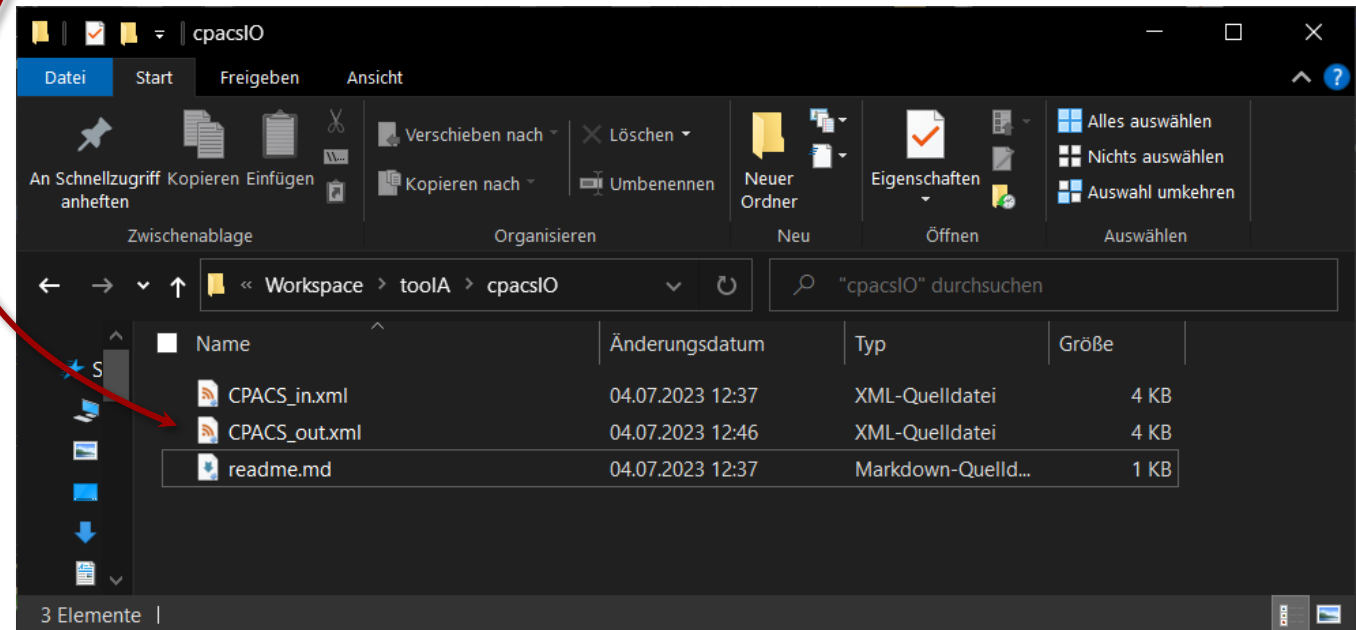
```
(base) C:\>conda activate cpacsEnv
(cpacsEnv) C:\>cd Workspace\toolA
(cpacsEnv) C:\Workspace\toolA>copy tests\testData\airports.xml cpacsIO\CPACS_in.xml
1 Datei(en) kopiert.
(cpacsEnv) C:\Workspace\toolA>_
```

Test your Python environment (2)

4. Run the python script *run.py* and check for the output *./cpacsIO/CPACS_out.xml*

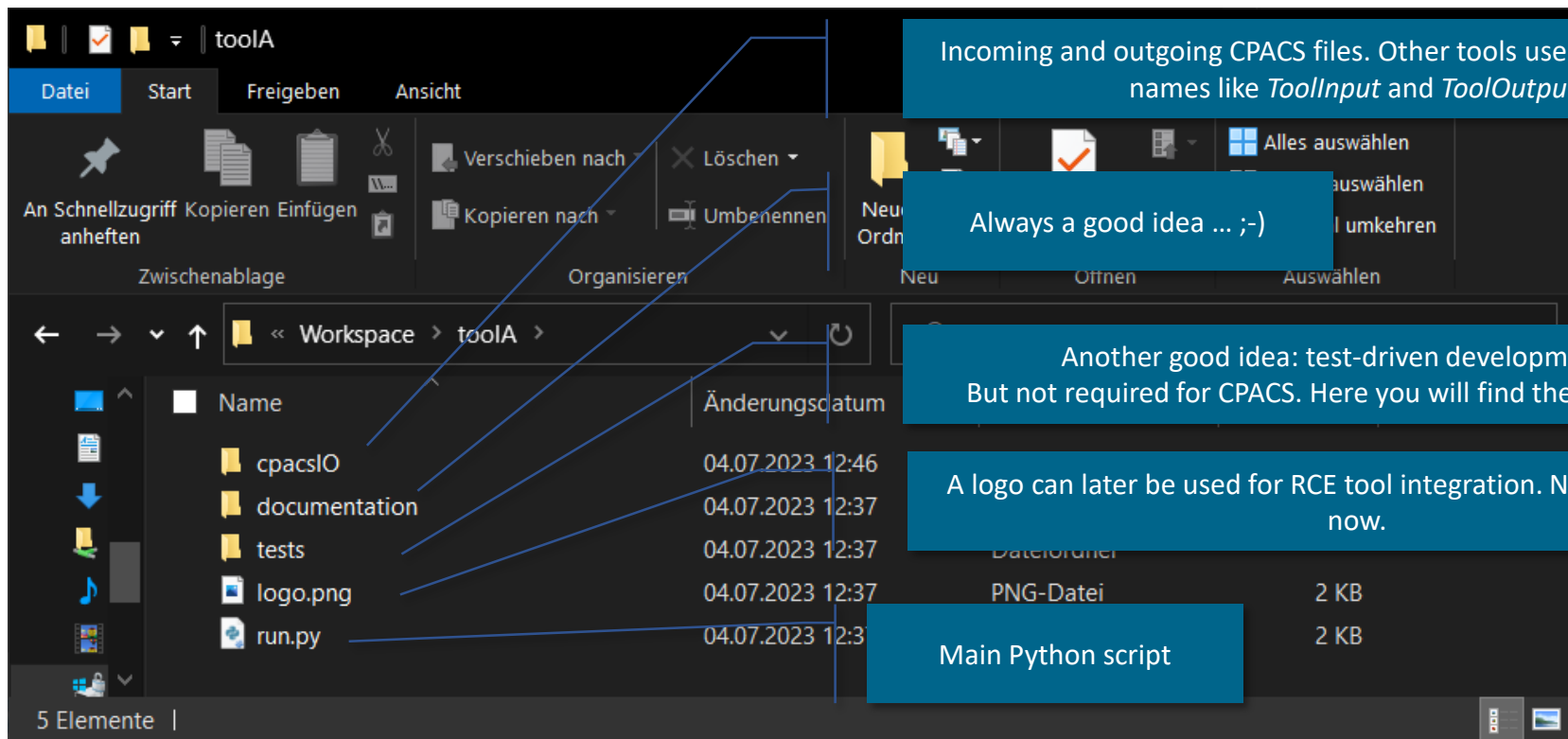
```
(cpacsEnv) C:\Workspace\toolA>python run.py
===== Welcome to ToolA =====
TIXI version: 3.3.0

(cpacEnv) C:\Workspace\toolA>
```



Test your Python environment (3)

- **Note:** We only request tool vendors to provide specific locations and file names for the incoming and outgoing CPACS files. Its your decision on how to define those. My example tool is structured like this:



Incoming and outgoing CPACS files. Other tools use similar directory names like *ToolInput* and *ToolOutput*.

Always a good idea ... ;-)

Another good idea: test-driven development ;-)
But not required for CPACS. Here you will find the example file.

A logo can later be used for RCE tool integration. Not important for now.

Main Python script

Name	Änderungsdatum	Dateigröße
cpacsIO	04.07.2023 12:46	
documentation	04.07.2023 12:37	
tests	04.07.2023 12:37	
logo.png	04.07.2023 12:37	2 KB
run.py	04.07.2023 12:37	2 KB

5 Elemente |

Understanding & adopting the Python script

- The philosophy behind this implementation:
 1. *Preprocessing*: Import CPACS file (e.g., *CPACS_in.xml*) and prepare the CPACS data for your tool
 2. *Compute*: Call your tool
 3. *Postprocessing*: Translate your tool data into CPACS and write a CPACS file (e.g., *CPACS_out.xml*)

(2) Subsequently call pre-processing, computation and postprocessing

(1) Typical Python notation to start executing code

Understanding & adopting the Python script

```
import os
import datetime
from tixi3 import tixi3wrapper
```

```
def preprocessing():
    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)
```

(3) Load TiXI library

(4) Use TiXI to open a CPACS file.

```
def compute():
    pass

def postprocessing():
    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:
        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changelog_xPath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"

        # Create new logEntry
        tixi_h.createElement(changelog_xPath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xPath = changelog_xPath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xPath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xPath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xPath, "creator", "ToolA")

    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()

def main():
    print('***** Welcome to ToolA *****')

    preprocessing()
    compute()
    postprocessing()

if __name__ == "__main__":
    main()
```

- TiXI offers comfortable reading and writing of CPACS files. All TiXI functions can be found at: <http://dlr-sc.github.io/tixi/>
- We could directly use *tixi_h* in our tool to have access to the CPACS data at any time (find all the available get-functions [here](#) or via *help(tixi_h)*)
- You can also use the preprocessing step to translate the data and write your tool input (e.g., via [csv APIs](#)).

Understanding & adopting the Python script

(5) Implement/run your tool

- Various approaches are possible here:
 - Directly implement your code here
 - Provide your tool as a module that can be imported in Python
 - Run your tool via command line from Python (e.g., `os.system()` or `subprocess.call()`; see [Tutorial](#))

```
import os
import datetime
from tixi3 import tixi3wrapper

def preprocessing():

    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)

def compute():
    pass

def postprocessing():

    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:

        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changelog_xPath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"

        # Create new logEntry
        tixi_h.createElement(changelog_xPath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xPath = changelog_xPath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xPath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xPath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xPath, "creator", "ToolA")

    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()

def main():

    print('===== Welcome to ToolA =====')

    preprocessing()
    compute()
    postprocessing()

if __name__ == "__main__":
    main()
```

Understanding & adopting the Python script

```
import os
import datetime
from tixi3 import tixi3wrapper

def preprocessing():

    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)

def compute():
    pass

def postprocessing():

    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:

        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changelog_xPath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"

        # Create new logEntry
        tixi_h.createElement(changelog_xPath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xPath = changelog_xPath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xPath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xPath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xPath, "creator", "ToolA")

    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()

def main():
    print('***** Welcome to ToolA *****')

    preprocessing()
    compute()
    postprocessing()

if __name__ == "__main__":
    main()
```

- Similar to preprocessing: Assume your tool ran successfully. Now you want to translate the data into CPACS and write the XML file.

Understanding & adopting the Python script

■ Exercise: add a new *logEntry* element:

```
import os
import datetime
from tixi3 import tixi3wrapper

def preprocessing():

    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)

def compute():

    pass

def postprocessing():

    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:

        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changelog_xPath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"

        # Create new logEntry
        tixi_h.createElement(changelog_xPath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xPath = changelog_xPath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xPath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xPath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xPath, "creator", "ToolA")

    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()

def main():

    print('===== Welcome to ToolA =====')

    preprocessing()
    compute()
    postprocessing()

if __name__ == "__main__":
    main()
```

```
<header>
  <name>Flight Path Example</name>
  <version>1.0</version>
  <versionInfos>
    <versionInfo version="1.0">
      <cpacsVersion>3.4-IM.0.1</cpacsVersion>
      <description>Flight Path from Frankfurt to Newark</description>
      <timestamp>2023-04-27T12:00:00</timestamp>
      <creator>Arthur Zamfir, DLR-SL</creator>
      <changeLog>
        <logEntry>
          <description>Created initial file</description>
          <timestamp>2023-04-27T12:00:00</timestamp>
          <creator>Arthur Zamfir, DLR-SL</creator>
        </logEntry>
        <logEntry>
          <description>Update to ImpactMonitoring project schema</description>
          <timestamp>2023-06-22T12:00:00</timestamp>
          <creator>M. Alder, DLR-SL</creator>
        </logEntry>
        <logEntry>
          <description>Added log for testing</description>
          <timestamp>2023-07-04T12:46:10</timestamp>
          <creator>ToolA</creator>
        </logEntry>
      </changeLog>
    </versionInfo>
  </versionInfos>
</header>
```

Understanding & adopting the Python script

```
import os
import datetime
from tixi3 import tixi3wrapper
```

```
def preprocessing():
```

```
    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)
```

```
def compute():
```

```
    pass

def postprocessing():
    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:

        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changeLog_xPath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"
```

```
        # Create new logEntry
        tixi_h.createElement(changeLog_xPath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xPath = changeLog_xPath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xPath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xPath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xPath, "creator", "ToolA")
```

```
    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()
```

```
def main():
```

```
    print('***** Welcome to ToolA *****')

    preprocessing()
    compute()
    postprocessing()
```

```
if __name__ == "__main__":
    main()
```

(5) We need to read the *version* of the data set in order to find the *versionInfo* node with the identical attribute

```
<header>
  <name>Flight Path Example</name>
  <version>1.0</version>
  <versionInfos>
    <versionInfo version="1.0">
      <cpacsVersion>3.4-IM.0.1</cpacsVersion>
      <description>Flight Path from Frankfurt to Newark</description>
      <timestamp>2023-04-27T12:00:00</timestamp>
      <creator>Arthur Zamfir, DLR-SL</creator>
      <changeLog>
        <logEntry>
          <description>Created initial file</description>
          <timestamp>2023-04-27T12:00:00</timestamp>
```

- There could be multiple *versionInfo* elements. We find the correct node by comparing the *version* attribute with the *version* of the data set.
- We use XPath to define the path in the CPACS data tree. Examples can be found at [w3schools](https://www.w3schools.com/xpath/).
- Note: *cpacsVersion* is the version of the CPACS schema, *version* is the version of the actual data set.

```
</versionInfo>
</versionInfos>
</header>
```


Understanding & adopting the Python script

```
import os
import datetime
from tixi3 import tixi3wrapper

def preprocessing():
    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)

def compute():
    pass

def postprocessing():
    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:
        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changeLog_xpath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"

        # Create new logEntry
        tixi_h.createElement(changeLog_xpath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xpath = changeLog_xpath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xpath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xpath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xpath, "creator", "ToolA")

    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()

def main():
    print('===== Welcome to ToolA =====')

    preprocessing()
    compute()
    postprocessing()

if __name__ == "__main__":
    main()
```

```
<header>
  <name>Flight Path Example</name>
  <version>1.0</version>
  <versionInfos>
    <versionInfo version="1.0">
      <cpacsVersion>3.4-TM 0.1</cpacsVersion>
      <description>Update to ImpactMonitoring project schema</description>
      <timestamp>2023-04-27T12:00:00</timestamp>
      <creator>Arthur Zampieri, DLR-SL</creator>
    </versionInfo>
  </versionInfos>
  <logEntry>
    <description>Update to ImpactMonitoring project schema</description>
    <timestamp>2023-06-22T12:00:00</timestamp>
    <creator>M. Alder, DLR-SL</creator>
  </logEntry>
  <logEntry>
    <description>Added log for testing</description>
    <timestamp>2023-07-04T12:46:10</timestamp>
    <creator>ToolA</creator>
  </logEntry>
</changeLog>
</versionInfo>
</versionInfos>
</header>
```

(7) Create a timestamp in the correct format (»YYYY-MM-DDThh:mm:ss«, see [docs](#))

(8) Define Xpath to last *logEntry* element

(9) Add *description*, *timestamp* and *creator* elements (see [docs](#))

Understanding & adopting the Python script

```
import os
import datetime
from tixi3 import tixi3wrapper

def preprocessing():

    # Load Tixi
    try:
        global tixi_h
        tixi_h = tixi3wrapper.Tixi3()
        print("TIXI version: ", tixi_h.version)
    except Exception as e:
        print(f"Could not load TIXI! Error message is: {e}")

    # Open CPACS file
    try:
        tixi_h.open('./cpacsIO/CPACS_in.xml')
    except Exception as e:
        print(e)

def compute():

    pass

def postprocessing():

    writeLogEntry = True

    # Add provenance information
    if writeLogEntry:

        # Current version of the dataset:
        version = tixi_h.getTextElement("/cpacs/header/version")
        changelog_xPath = f"/cpacs/header/versionInfos/versionInfo[@version='{version}']/changeLog"

        # Create new logEntry
        tixi_h.createElement(changelog_xPath, "logEntry")

        timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

        logEntry_xPath = changelog_xPath + "/logEntry[last()]"
        tixi_h.addTextElement(logEntry_xPath, "description", "Added log for testing")
        tixi_h.addTextElement(logEntry_xPath, "timestamp", timestamp)
        tixi_h.addTextElement(logEntry_xPath, "creator", "ToolA")

    # Write CPACS file
    cpacsOut = os.path.join(os.getcwd(), 'cpacsIO', 'CPACS_out.xml')
    tixi_h.save(cpacsOut)
    tixi_h.close()

def main():

    print('===== Welcome to ToolA =====')

    preprocessing()
    compute()
    postprocessing()

if __name__ == "__main__":
    main()
```

(10) Write data to XML file: In this example *CPACS_out.xml* (see [docs](#))

Questions? Confused?



- marko.alder@dlr.de
- I'm happy to support you with a bilateral exchange.



IMPACT MONITOR

Thank you!



Funded by
the European Union



Coordinated by
the German Aerospace Center



marko.alder@dlr.de

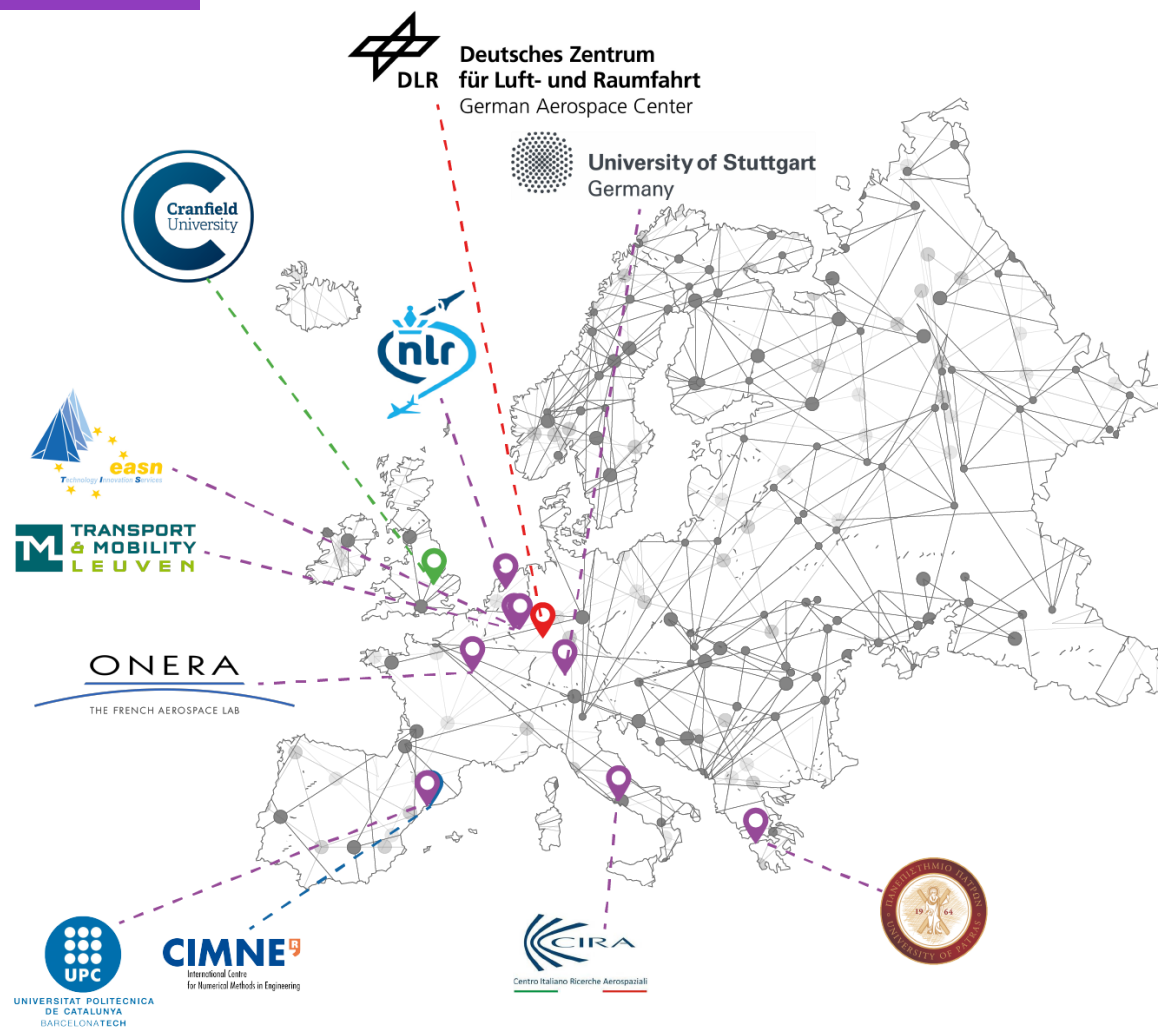


www.dlr.de



DLR Hamburg

The team



Acknowledgments



Funded by
the European Union

Funded by the European Union under GA No. 101097011.

Views and opinions expressed are however those of the author(s) only and not necessarily reflect those of the European Union or CINEA. Neither the European Union nor CINEA can be held responsible for them.

This document and its contents remain the property of the beneficiaries of the Impact Monitor Consortium. It may contain information subject to intellectual property rights. No intellectual property rights are granted by the delivery of this document or the disclosure of its content. Reproduction or circulation of this document to any third party is prohibited without the consent of the author(s).



IMPACT MONITOR



Funded by
the European Union



Coordinated by
the German Aerospace Center

