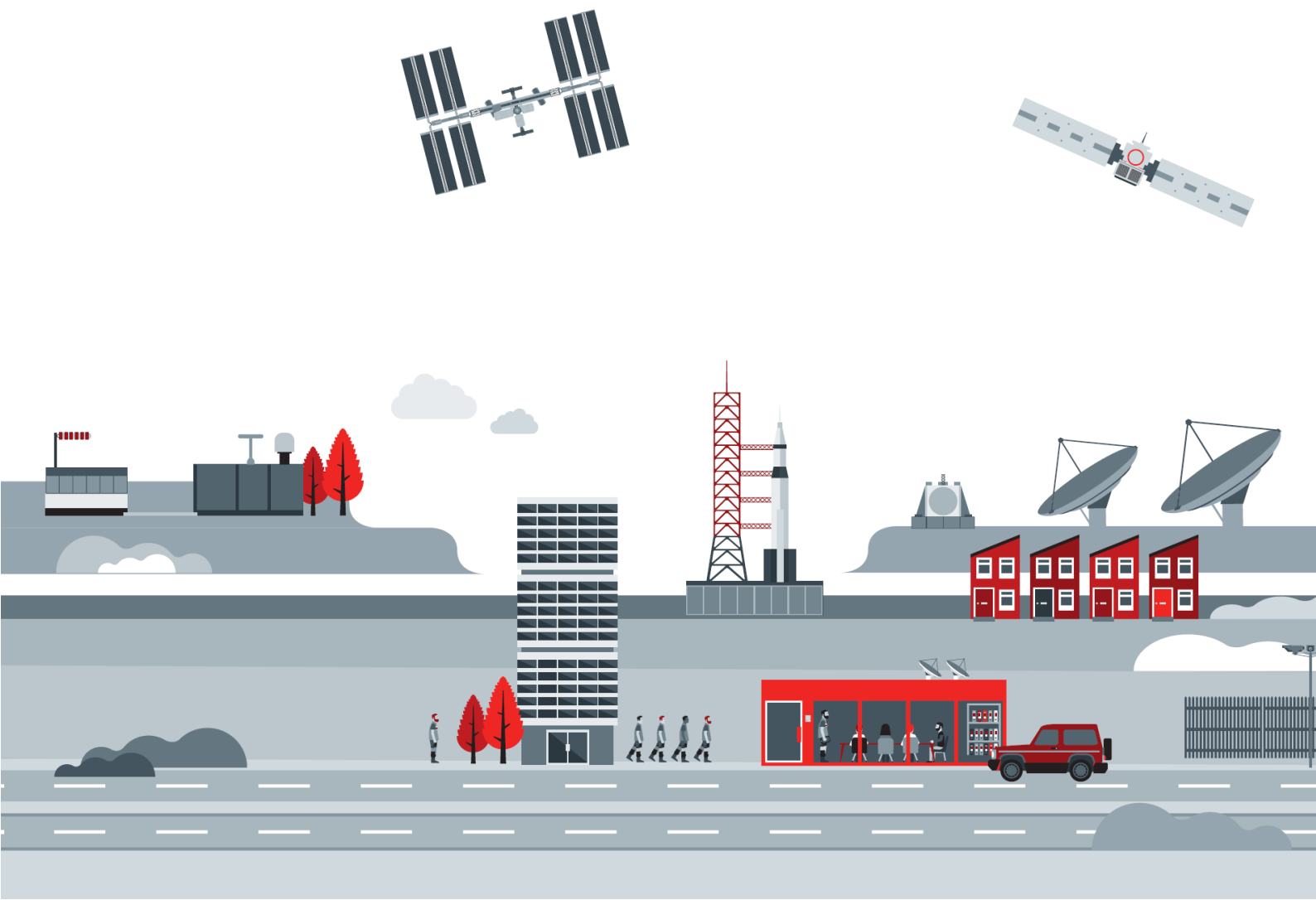
Collaborative Data Hub Software - Maintenance and Evolution Services - Ready for Digital Twin Earth

Transformation Framework
Installation and Configuration Manual



Reference: COPE-SERCO-TN-21-1218
Date: 26/10/2023
Version: 4.1

Role/Title	Name	Signature	Date
Author	Giuseppe Le Voci, Aureliana Barghini, Luca Fabbri, Marco Di Bari		11/09/2023
Verified	Barbara Borgia		11/09/2023
Approved	Andrea Tesseri		11/09/2023

Change register

Version/Rev.	Date	Change	Reason
1.0	30/11/2021		First release of the document
1.1	15/12/2021	Updated installation instructions & minor fixes	Align to software README
1.2	17/12/2021	Updated installation instructions to download code from public repository	Public repository available
1.3	02/03/2022	Upadated configuration of the TF with Keycloak and a proxy	Realease v0.8.2-osf
1.4	08/04/2022	Aligned docker-compose setup to latest updates Add details on role, keeping period and exclude_workflows configuration Add plugin installation Add NGINX access log	Release v0.9

1.5	14/04/2022	add description of OUTPUT_GROUP_OWNER_ID add description how to define role in config file	Release esa_tf v0.9.1
1.6	01/08/2022	Add traceability configuration	Release esa_tf v1.1.1-osf
2.1	03/02/2023	Add monitoring	Release esa_tf v1.3.0-osf
3.0	11/04/2023	Add configuration of csc-api data-sources	Release esa_tf v1.4.0-osf
4.0	06/09/2023	Add EOP plugin description Add parallel processing configuration	Release esa_tf v1.5.2-osf
4.1	26/10/2023	Link to GitHub in section 2.1 updated	Update of 4.0 Installation Manual

Table of Contents

1. Introduction

1.1 Scope

This document reports the instructions to install, configure and use the Transformation Framework within the Collaborative Data Hub Software Maintenance and Evolution Services for Digital Twin Earth. Installation and configuration steps may serve Collaborative administrators and eventual developers of the framework. Usage instructions address all TF users.

1.2 Purpose

This document aims to detail step-by-step instructions to install the Transformation Framework. It subsequently seeks to provide the necessary information concerning the current features offered by the Framework, including details on how to tailor the configuration of the sen2cor plugin to match user needs, finally delivering short examples on how to perform requests to the framework and trigger transformation orders.

1.3 Document applicability

Please note that this document is referring to version v1.5.2-osf, belonging to the DHS#6 release.

Component	Component Release	DHS Release
Transformation Framework	v1.5.2-osf	DHS#6

1.4 Document structure

The document is structured as follows:

- Section 1 (this section): Introduction, providing document structure, reference documents and definitions/acronyms.
- Section 2 contains installation instructions for the Transformation Framework.
- Section 3 provides details on the configuration of the TF.
- Section 4 details usage examples for transformation and workflow order requests.

1.5 Applicable Documents

Ref.	Title	Reference and Version
AD-1	[AD-SOW] Statement of Work: COLLABORATIVE DATA HUB SOFTWARE - MAINTENANCE AND EVOLUTION SERVICES - READY FOR DIGITAL TWIN EARTH	ESA-EOPG-EOPGC-SOW-12, v1.0
AD-2	DHS System Design Document	COPE-SERCO-TN-21-1171
AD-3	[AD-PRIP] Production Interface Delivery Point Specification - ICD	ESA-EOPG-EOPGC-IF-3, v1.6

Ref.	Title	Reference and Version
AD-4	Collaborative Data Hub Software – Transformation Framework – ICD	COPE-SERCO-IF-21-1178, v4.0
AD-5	SRTM DEM download path	http://srtm.csi.cgiar.org/wp-content/uploads/files/srtm_5x5/TIFF/
AD-6	Transformation Framework SRS	COPE-SERCO-RD-21-1177 - Collaborative Data Hub Software - SRS v2.3
AD-7	Collaborative Data Hub Software – Evolution Roadmap	COPE-SERCO-PL-21-1158, v1.1

1.6 Reference Documents

Ref.	Title	Reference and Version
RD-1.	HTTP Status Codes	<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
RD-2.	[AD-TR-TN] Technical Note for the Interfacing of the Traceability Service by Production Services in RFP-1	ESA-EOPG-EOPGC-TN-23, v.TBD
RD-3.	[AD-TR-ICD] Traceability Service Interface Control Document	ESA-EOPG-EOPGC-IF-9, Issue 1/2
RD-4.	Dask.distributed lightweight library for distributed computing in Python	<https://distributed.dask.org>
RD-5.	Python entry-points specifications	<https://packaging.python.org/specifications/entry-points/>
RD-6.	Sen2cor plugin v2.9 – User guide	<http://step.esa.int/thirdparties/sen2cor/2.9.0/docs/S2-PDGS-MPC-L2A-SUM-V2.9.0.pdf>
RD-7.	[AS-TR-UM] GAEL-P299-SUM-001-01-10 - Traceability Service User Manual	V1.10

1.7 Acronyms and Abbreviations

Acronym	Definition
AD	Applicable Document
API	Application Programming Interface
DHS	Data Hub Software
DHuS	Data Hub Service
EO	Earth Observation
ESA	European Space Agency
GS	Ground Segment
HTTP	Hypertext Transfer Protocol Secure
HTTPS	Hypertext Transfer Protocol Secure

ICD	Interface Control Document
OData	Open Data Protocol
RD	Reference Document
SOW	Statement of Work
TF	Transformation Framework
UUID	Universally unique identifier

2. Transformation Framework Installation

The ESA Transformation Framework is a component of the Copernicus Sentinels Collaborative Data Hub Software (DHS) intended to provide data transformation capabilities via the integration of processing elements applied on-demand to Copernicus Sentinel products, prior to delivery to the users.

2.1 Docker compose setup

Installation of the TF takes place via docker-compose (see the Overview section in [AD-4] for more information). The following software must be pre-installed on the Virtual Machine:

- `docker engine (>=20.10)`
- `docker compose (>=2.0)`
- `make (>=3.81)`
- `curl (>=7.75)`
- `unzip (>=6.0)`
- `tar (>=2.8)`

Memory and size resources required to setup the framework must be specified in the configuration of the docker engine as follows:

- memory > 6Gb per core
- disk image size > 50Gb

Specifically, these requirements are mandatory in order to submit transformations using sen2cor plugin, the only plugin available in version v1.5.2-osf of the Framework.

When the download completes, open a terminal, navigate to the directory where the installation will take place and execute the following commands to download the .zip file containing the source code from the public repository on GitHub, uncompressed the file, navigate to the directory of the Transformation Framework code containing the Makefile with the setup instructions, and download the external resources required by the framework:

```
curl -https://github.com/DHS-TransformationFramework/esa_tf/archive/refs/tags/v.1.5.2-osf.zip

unzip esa_tf-v.1.5.2-osf.zip
cd esa_tf-v.1.5.2-osf/esa_tf
make setup
```

In order to start the docker compose, it is then necessary to configure user names and passwords which allow access to the external data sources, in file `config/hubs_credentials.yaml`.

In a dedicated `.env` file in the `esa_tf` folder, the following environment variables can be configured:

- `OUTPUT_OWNER_ID`: the id of the owner of the output files
- `OUTPUT GROUP_OWNER_ID:` the group id of the owner of the output files
- `OUTPUT_DIR`: the output folder where the TF output files will be saved.

- PLUGINS_DIR: a folder where users must load the Python wheels corresponding to any additional custom plugins they may want to install. The plugins will be installed automatically at container startup.

Finally, the following command pulls the docker image from the registry located in the ONDA DIAS repository and starts the docker compose:

```
make up
```

Note: The version that will be downloaded is defined in the esa_tf/.env file with the parameter ESA_TF_RELEASE and corresponds to the esa_tf software version (in this case v1.5.2-osf), but can be changed. If the parameter ESA_TF_RELEASE is commented out, the TF will always use the latest version.

2.2 Test the Installation

The API endpoints will now be available at http://<IP>:<PORT>, assuming the TF is exposed Note: the version that will be downloaded is defined in the esa_tf/.env file with the parameter ESA_TF_RELEASE and corresponds to the esa_tf software version (in this case v1.5.2-osf), but can be changed. If the parameter ESA_TF_RELEASE is commented out, the TF will always use the latest version.at the IP address, and the PORT is defined in the docker-compose.

The following URL can then be visited to display the list of available plugins:

```
http://<IP>:<PORT>/Workflows
```

Alternatively, the following software can be used to test REST API endpoints from a command line interface:

```
curl
jq
```

The equivalent command in order to display the plugins list would then be:

```
curl http://<IP>:<PORT>/Workflows | jq
```

The corresponding expected payload is:


```
{
  "value": [
    {
      "Id": "eopf_convert_to_cog",
      "WorkflowName": "eopf_convert_to_cog",
      "Description": "EOPF plugin for converting Sentinel-1, Sentinel-2 and Sentinel-3 SAFE in COG
format",
      "InputProductType": [
        "WV_SLC__1S",
        "IW_SLC__1S",
        "EW_SLC__1S",
        "IW_GRDH_1S",
        "EW_GRDH_1S",
        "IW_GRDM_1S",
        "EW_GRDM_1S",
        "IW_OCN__2S",
        "EW_OCN__2S",
        "S2MSI1C",
        "S2MSI2A",
```

```
        "SR_1_SRA_BS",
        "SL_1_RBT___",
        "OL_1_EFR___",
        "OL_1_ERR___",
        "OL_2_LRR___",
        "SL_2_LST___",
        "SL_2_FRP___",
        "SY_2_SYN___",
        "SR_2_LAN___"
      ],
      "OutputProductType": null,
      "WorkflowVersion": "0.1",
      "WorkflowOptions": {
        "cog_compression": {
          "Description": "Type of compression",
          "Type": "string",
          "Default": "DEFLATE",
          "Enum": [
            "NONE",
            "LZW",
            "JPEG",
            "DEFLATE",
            "ZSTD",
            "WEBP",
            "LERC",
            "LERC_DEFLATE",
            "LERC_ZSTD",
            "LZMA"
          ]
        }
      }
    },
    {
      "Id": "eopf_convert_to_netcdf",
      "WorkflowName": "eopf_convert_to_netcdf",
      "Description": "EOPF plugin for converting Sentinel-1, Sentinel-2 and Sentinel-3 SAFE in netcdf
format",
      "InputProductType": [
        "IW_OCN__2S",
        "EW_OCN__2S",
        "S2MSI1C",
        "S2MSI2A",
        "SR_1_SRA_BS",
        "SL_1_RBT___",
        "OL_1_EFR___",
        "OL_1_ERR___",
        "OL_2_LRR___",
        "SL_2_LST___",
        "SL_2_FRP___",
        "SY_2_SYN___",
        "SR_2_LAN___"
      ],
      "OutputProductType": null,
      "WorkflowVersion": "0.1",
      "WorkflowOptions": {
        "netcdf_compression": {
          "Description": "Activate the compression",
          "Type": "boolean",
          "Default": true,
          "Enum": [
            true,
            false
          ]
        },
```

```
        "netcdf_comp_level": {
          "Description": "Compression level",
          "Type": "integer",
          "Default": 1,
          "Enum": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9
          ]
        },
        "netcdf_shuffle": {
          "Description": "Activate the shuffle",
          "Type": "string",
          "Default": "YES",
          "Enum": [
            "YES",
            "NO"
          ]
        }
      }
    },
    {
      "Id": "eopf_convert_to_zarr",
      "WorkflowName": "eopf_convert_to_zarr",
      "Description": "EOPF plugin for converting Sentinel-1, Sentinel-2 and Sentinel-3 SAFE in zarr
format",
      "InputProductType": [
        "WV_SLC__1S",
        "IW_SLC__1S",
        "EW_SLC__1S",
        "IW_GRDH_1S",
        "EW_GRDH_1S",
        "IW_GRDM_1S",
        "EW_GRDM_1S",
        "IW_OCN__2S",
        "EW_OCN__2S",
        "S2MSI1C",
        "S2MSI2A",
        "SR_1_SRA_BS",
        "SL_1_RBT___",
        "OL_1_EFR___",
        "OL_1_ERR___",
        "OL_2_LRR___",
        "SL_2_LST___",
        "SL_2_FRP___",
        "SY_2_SYN___",
        "SR_2_LAN___"
      ],
      "OutputProductType": null,
      "WorkflowVersion": "0.1",
      "WorkflowOptions": {
        "dask_compression": {
          "Description": "Type of compression",
          "Type": "string",
          "Default": "zstd",
          "Enum": [
            "zstd",
            "blosclz",
```

```
            "lz4",
            "lz4hc",
            "zlib",
            "snappy"
          ]
        },
        "dask_comp_level": {
          "Description": "Compression level",
          "Type": "integer",
          "Default": 1,
          "Enum": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9
          ]
        },
        "dask_shuffle": {
          "Description": "Shuffle: NOSHUFFLE (0), SHUFFLE (1), BITSHUFFLE (2) or AUTOSHUFFLE (-1)",
          "Type": "integer",
          "Default": 2,
          "Enum": [
            0,
            1,
            2,
            -1
          ]
        }
      }
    },
    {
      "Id": "sen2cor_l1c_l2a",
      "WorkflowName": "Sen2Cor_L1C_L2A",
      "Description": "Product processing from Sentinel-2 L1C to L2A using Sen2Cor v2.10, supporting
Level-1C product version 14.2 - 14.9",
      "InputProductType": "S2MSI1C",
      "OutputProductType": "S2MSI2A",
      "WorkflowVersion": "0.2",
      "WorkflowOptions": {
        "Aerosol_Type": {
          "Description": "Default processing via configuration is the rural (continental) aerosol type
with mid latitude summer and an ozone concentration of 331 Dobson Units",
          "Type": "string",
          "Default": "RURAL",
          "Enum": [
            "MARITIME",
            "RURAL"
          ]
        },
        "Mid_Latitude": {
          "Description": "If 'AUTO' the atmosphere profile will be determined automatically by the
processor, selecting WINTER or SUMMER atmosphere profile based on the acquisition date and geographic
location of the tile",
          "Type": "string",
          "Default": "SUMMER",
          "Enum": [
            "SUMMER",
            "WINTER",
            "AUTO"
```

```
          ]
        },
        "Ozone_Content": {
          "Description": "0: to get the best approximation from metadata (this is the smallest
difference between metadata and column DU), else select for midlatitude summer (MS) atmosphere: 250,
290, 331 (standard MS), 370, 410, 450; for midlatitude winter (MW) atmosphere: 250, 290, 330, 377
(standard MW), 420, 460",
          "Type": "integer",
          "Default": 331,
          "Enum": [
            0,
            250,
            290,
            330,
            331,
            370,
            377,
            410,
            420,
            450,
            460
          ]
        },
        "Cirrus_Correction": {
          "Description": "FALSE: no cirrus correction applied, TRUE: cirrus correction applied",
          "Type": "boolean",
          "Default": false,
          "Enum": [
            true,
            false
          ]
        },
        "DEM_Terrain_Correction": {
          "Description": "Use DEM for Terrain Correction, otherwise only used for WVP and AOT",
          "Type": "boolean",
          "Default": true,
          "Enum": [
            true,
            false
          ]
        },
        "Resolution": {
          "Description": "Target resolution, can be 10, 20 or 60m. If omitted, 10, 20 and 60m
resolutions will be processed",
          "Type": "integer",
          "Default": null,
          "Enum": [
            10,
            20,
            60
          ]
        }
      }
    }
  ]
}
```


2.3 Plugins installation

The installation of a new plugin requires two steps:

- Create the plugin wheel.
- Load the wheel in the {PLUGINS_DIR} folder (that can be defined in the `esa_tf/.env` file)

The plugins inside the {PLUGINS_DIR} folder will be installed automatically at container startup.

3. Transformation Framework Configuration

The Transformation Framework is presently available in its version 1.5.2-osf. This Section draws a map of the current features and the configuration of the first plugin available.

3.1 Service Installation Requirements

The TF core is designed to be used locally by any user with simple commands and minimal requirements. When installed as a service on a production environment, it is required to interact with external services like Web proxy and IAM services.

3.1.1 Publish the TF behind a proxy

Although the TF provides an internal proxy layer, it is intended for internal use only (logging and download management). Other tasks, like using an official domain or adding SSL protection should be provided by an external front-end proxy.

There are two tasks that should be properly configured:

- Publish the TF on an official domain (with HTTPS/SSL)
- Publish the TF in a sub-path of the same domain

As example: let say we want to publish our TF installation on https://mynetwork.net/transformation-framework

3.1.1.1 Configuring the domain (and SSL)

The internal proxy layer is already configured to receive and pass-through proper HTTP headers from the outer front-end proxy.

The minimal configuration required by the front-end proxy is to pass a `Host` header, as it will be used to compose the hostname of generated URLs.
In our example: this Host header will be `mynetwork.net`.
If the front-end proxy provides SSL (which is strongly recommended), the `X-Forwarded-Proto` header should be also set.
In our example we are using HTTPS/SSL, so our `X-Forwarded-Proto` should use `https`.

Other `X- Forwarded -*` headers can also be useful for improving HTTP logging.

In case changing the configuration of the front-end proxy is not applicable, it is possible to forcibly set two environment variables:

- APPLICATION_HOSTNAME, default: empty string. It is the hostname where the TF will be published. Instead of setting this value, you can configure your front-end proxy to properly set the `Host` HTTP header. In our example it will be again "`mynetwork.net`"
- APPLICATION_PROTO, default: empty string. It is the HTTP protocol used to serve the TF. Instead of setting this value, you can configure your front-end proxy to properly set the `X-Forwarded-Proto` HTTP header. In our example: "`https`"

Please note: using any APPLICATION_* environment variables above will override any headers received by the external proxy.

3.1.1.2 Configuring a subpath

The internal proxy is configured to serve the TF at root level.
For example: access to the list of transformations is performed by calling:

```
GET http://mynetwork.net/TransformationOrders.
```

When publishing the TF on the network an entry prefix path may be preferred, such as:

```
GET http://mynetwork.net/transformation-framework/TransformationOrders.
```

In this case you should instruct the TF about the prefix by passing a ROOT_PATH environment variable.
In our example:

```
ROOT_PATH=/transformation-framework
```

3.1.1.3 Network security

1. By default, the entry point of the TF is an HTTP service running on port 8080, but many other ports are available for debugging purpose:
   - 8000: OData RESTful API endpoint
   - 8786: Dask scheduler
   - 8787: Dask diagnostic service
2. These ports of the Docker host must not be opened to external access on a production system.

3.1.2 Keycloak integration

Integration with Keycloak is performed from the internal NGINX proxy.
Keycloak should provide a client specifically configured for the TF, configured as follows:

- Client protocol: openid-connect
- Access Type: bearer-only

This first client will only validate access token but it's not allowed to generate them. For this reason a second Keycloak client will be required with the following configuration:

- Client protocol: openid-connect
- Access Type: public

Authentication works as follow:

- The user must obtain an access_token from public Keycloak client. How the user obtains the token (using curl, postman or any other REST application) is out of the scope of the TF.
- With this access_token, the user will be able to query the TF providing an Authorization header.

Keycloak is disabled by default. It must be enabled and configured using environment variables:

- `OIDC_ACTIVE`: set to true to enable Keycloak integration
- `OIDC_ROOT_URL`: required - URL to the Keycloak service (the NGINX proxy must be able to reach this URL)
- `REALM_NAME`: required - Keycloak realm name

- `CLIENT_ID`: required - Keycloak client id
- `CLIENT_SECRET`: required - Keycloak client secret
- GUARD_ROLE: optional - use this configuration to protect the TF from users without a role (this will commonly be an automatic role from the client). The format should be `<client_id>:<role>`.
  If not provided, every authenticated user in the realm will be able to access the TF.

3.2 Data sources configuration

The data source shall be configured in the `${CONFIG_DIR}/hubs_credentials.yaml` file, where the `${CONFIG_DIR}` can be defined in esa_tf/.env file.

In `${CONFIG_DIR}/hubs_credentials.yam, it` shall be defined:

- api_type: that can be `dhus-api or csc-api`
- the credentials

in case the api_type is `csc-api`, it shall be defined also:

- `auth`, that is the authentication type and it can be oauth2 or basic
- `query_auth:` define if the authentication is needed or not for querying the data source. It can be True or False
- `download_auth:` define if the authentication is needed or not for downloading the data from the source. It can be True or False

The following snippet represents an example of configuration file:

```
apihub:
  api_type: dhus-api
  credentials:
    api_url: https://apihub.copernicus.eu/dhus
    user: my-apihub-username
    password: my-apihub-password
scihub:
  api_type: dhus-api
  credentials:
    api_url: https://scihub.copernicus.eu/dhus
    user: my-scihub-username
    password: my-scihub-password
cdse:
  api_type: csc-api
  auth: oauth2
  query_auth: False
  download_auth: True
  credentials:
    api_url: https://catalogue.dataspace.copernicus.eu
    user: my-cdse-username
    password: my-cdse-password
    token_endpoint: https://identity.dataspace.copernicus.eu/auth/realms/CDSE/protocol/openid-connect/token
    client_id: cdse-public
gss:
  api_type: csc-api
  auth: basic
  query_auth: True
  download_auth: True
  credentials:
    api_url: https://dhs-test.onda-dias.eu/gss-cat-01
    user: my-gss-username
    password: my-gss-password
```

The TF will generate an example of `${CONFIG_DIR}/hubs_credentials.yaml`, running in the folder `esa_tf` the command:
`make setup`

3.3 Roles configuration

Quotas and profiles shall be configured in the YAML file, `${CONFIG_DIR}/esa_tf.config`, where the `${CONFIG_DIR} folder` can be defined in `esa_tf/.env` file.

By running in folder `esa_tf` the command `make setup`, the TF will generate an example config file.

The quota is the maximum number of TransformationOrders in queued status or in processing status for each user.
The profiles are related to the authorizations. Currently, two profiles are available: `user` and `manager`.

The user profile will be able to:

- request the configuration of all available workflows or a specific configuration
- submit a TransformationOrder
- request the list of TransformationOrders or one specific transformation

The manager profile will be able also to request the list of all the transformations submitted.

The check on the quota and on the profile can be disabled in the configuration file.

The following snippet represents an example of configuration. Note that “role” inside the TF shall be always a couple composed as “`<client_id>:<role_name>`”:

```
# enable authorization check
enable_authorization_check: false

# enable quota configuration
enable_quota_check: true

# role configuration
# - quota is maximum number of transformation orders in progress or queued per user
# - profile can ben "manager", "user"

# default role configuration
default_role:
  quota: 2
  profile: user

roles:
  “client_id:tf_manager”:
    quota: 10
    profile: manager
```

In this example, the `tf_manager` role has a `manager profile` and `quota` 10.

Since `tf_manager` is the only role defined in the configuration file, for all the users that do not have the `tf_manager` role, the TF will use the `default_role`. Note that `default_role` can be customized and, if removed from the configuration file, the TF will use its internal defaults: `profile user` and `quota` 2.

3.4 Keeping period and excluded workflows

The keeping period and the list of workflows to be excluded shall be defined in the YAML file `${CONFIG_DIR}/esa_tf.config`, where the `${CONFIG_DIR}` can be defined in esa_tf/.env file.

The TF will generate an example of configuration file running in the folder `esa_tf` the command: `make setup`

The `keeping_period` shall be an integer. It represents the minimum number of minutes from the CompletedDate for which a TransformationOrder will be kept in memory. The old TransformationOrders are deleted when a new transformation is requested.

The following snippet is an example of configuration file:

```
keeping-period: 14400
exclude-workflow:
  - workflow1_id
  - workflow2_id
```

3.5 Trace Sender Configuration

The Trace Sender is in charge to generate the trace and pushing it to the Traceability Services (TS), for more details see [AS-TR-UM] GAEL-P299-SUM-001-01-10 - Traceability Service User Manual RD-7.

The trace generation is done by Transformation Framework using trecetool-1.2.4 available at https://repository.gael-systems.com/repository/public/fr/gael/datac/tracetool/1.2.4/trecetool-1.2.4-distribution.

In order to push a trace, it is necessary:

- Request a TS user account, that will provide a username and a password
- Request TS Data Producer Role for the desired service context, e.g. `colgs` (see [RD-7] paragraph 5.3). With this role, you will be provided with a key fingerprint, passphrase and secret key file relative to the context of the required services.
- The secret key file must be renamed `secret.txt` and put in the `config` directory, while `traceability.config` YAML file in the config directory shall contain the TS services URL, the TS URL access token, the credentials obtained and the services information, for more details see [**Errore. L'origine riferimento non è stata trovata.**].

In the following an example of configuration file:

```
service_url: https://demo.trace.gael-systems.com
url_access_token: https://demo.trace.gael-
systems.com/auth/realms/ts/protocol/openid-connect/token
url_push_trace: https://demo.trace.gael-systems.com/trace-api/Traces
username: my-traceability-username
```

```
password: my-traceability-password
key_fingerprint: my-key-fingerprint
passphrase: my-key-passphrase
service_context: my-service
service_provider: my-service-provider
```

The trace sender can be activated or deactivated using the key `enable_traceability` in `esa_tf.config` YAML file file in `esa_tf/config` folder. In the same file, with the key `untraced_workflow`, each individual workflow can be disabled.

If the trace push fails, then the trace will be stored in the folder {TRACES_DIR} that can be configured in the `.env file` (by default it is ./`traces`) contained in `esa_tf/config folder`.

3.6 Resource Monitor Configuration

The resource monitor component is in charge of computing and logging the resources used by each Transformation order: total CPU usage, RAM usage, disk usage, wall time.

The Resource Monitor can be activated or deactivated using the key `enable_monitoring` in `esa_tf.config` YAML file in `esa_tf/config` folder. In `esa_tf.config`, you can also configure the monitoring_polling_period_s, i.e. the time interval used for sampling disk usage, subprocess CPU usage and RAM usage.

3.7 Sen2cor Plugin Configuration

The sen2cor_l1c_l2a workflow plugin installed within the TF makes use of Sen2Cor v2.10 tool to convert Sentinel-2 L1C products into L2A output product. It implements classification and atmospheric correction by using the Digital Elevation Model of the Shuttle Radar Topography Mission (*SRTM DEM)*. The model is downloaded automatically from [AD.5].

When submitting a request for the Transformation of a Sentinel-2 L1C product into a L2A product, the user can modify the configuration of the plugin by passing in various desired options. Assuming the TF has been installed following the instructions in Section 2, the user is given the ability to interrogate the TF on which options are accepted by the plugin, as its definition is accessible. It is possible to retrieve the definition of all plugins via the following URL:

`http://<IP>:<PORT>/Workflows`

As in future versions more plugins will be available, the sole configuration of the sen2cor_l1c_l2a plugin is retrievable via the following URL, specifying the plugin name as argument:

`http://<IP>:<PORT>/Workflows('sen2cor_l1c_l2a')`

The current definition of the plugin will be returned as follows:

```
{
  "@odata.id": "http://<IP>:<PORT>/Workflows('sen2cor_l1c_l2a')",
  "Id": "sen2cor_l1c_l2a",
  "WorkflowName": "Sen2Cor_L1C_L2A",
  "Description": "Product processing from Sentinel-2 L1C to L2A using Sen2Cor v2.10, supporting Level-
1C product version 14.2 - 14.9",
  "InputProductType": "S2MSI1C",
  "OutputProductType": "S2MSI2A",
```

```
  "WorkflowVersion": "0.2",
  "WorkflowOptions": {
    "Aerosol_Type": {
      "Description": "Default processing via configuration is the rural (continental) aerosol type with
mid latitude summer and an ozone concentration of 331 Dobson Units",
      "Type": "string",
      "Default": "RURAL",
      "Enum": [
        "MARITIME",
        "RURAL"
      ]
    },
    "Mid_Latitude": {
      "Description": "If 'AUTO' the atmosphere profile will be determined automatically by the
processor, selecting WINTER or SUMMER atmosphere profile based on the acquisition date and geographic
location of the tile",
      "Type": "string",
      "Default": "SUMMER",
      "Enum": [
        "SUMMER",
        "WINTER",
        "AUTO"
      ]
    },
    "Ozone_Content": {
      "Description": "0: to get the best approximation from metadata (this is the smallest difference
between metadata and column DU), else select for midlatitude summer (MS) atmosphere: 250, 290, 331
(standard MS), 370, 410, 450; for midlatitude winter (MW) atmosphere: 250, 290, 330, 377 (standard MW),
420, 460",
      "Type": "integer",
      "Default": 331,
      "Enum": [
        0,
        250,
        290,
        330,
        331,
        370,
        377,
        410,
        420,
        450,
        460
      ]
    },
    "Cirrus_Correction": {
      "Description": "FALSE: no cirrus correction applied, TRUE: cirrus correction applied",
      "Type": "boolean",
      "Default": false
    },
    "DEM_Terrain_Correction": {
      "Description": "Use DEM for Terrain Correction, otherwise only used for WVP and AOT",
      "Type": "boolean",
      "Default": true
    },
    "Resolution": {
      "Description": "Target resolution, can be 10, 20 or 60m. If omitted, 10, 20 and 60m resolutions
will be processed",
      "Type": "integer",
      "Enum": [
        10,
        20,
        60
      ],
```

```
      "Default": null
    }
  }
}
```

The user can then submit a processing request as explained in Section 4.2 below. For each process, the plugin initially verifies the validity of the options passed in by the user, then performs a consistency check in order to verify a valid Sentinel-2 L1C product has been fed to the TF. The default sen2cor configuration file `L2A_GIPP.xml` is then parsed, and the processing options eventually specified by the user are substituted within the file. The new configuration is saved in a temporary file which is read during the processing, and removed once the output file has been created and the processing finalized.

The available workflow options for the plugin are summarized in the following table:

Parameter	Allowed Values	Default Values	Description
Aerosol_Type	MARITIME, RURAL	RURAL	Default processing via configuration is the rural (continental) aerosol type with mid latitude summer and an ozone concentration of 331 Dobson Units.
Mid_Latitude	SUMMER, WINTER, AUTO	SUMMER	If 'AUTO' the atmosphere profile will be determined automatically by the processor, selecting WINTER or SUMMER atmosphere profile based on the acquisition date and geographic location of the tile
Ozone_Content	[0, 250, 290, 330, 331, 370, 377, 410, 420, 450, 460]	331	0: to get the best approximation from metadata (this is the smallest difference between metadata and column DU), else select for midlatitude summer (MS) atmosphere: 250, 290, 331 (standard MS), 370, 410, 450; for midlatitude winter (MW) atmosphere: 250, 290, 330, 377 (standard MW), 420, 460
Cirrus_Correction	True, False	True	FALSE: no cirrus correction applied, TRUE: cirrus correction applied
DEM_Terrain_Correction	True, False	True	Use DEM for Terrain Correction, otherwise only used for WVP and AOT
Resolution	[10, 20, 60]	Omit the parameter for Default Behaviour	Target resolution can be 10, 20 or 60m. The Resolution parameter could also be omitted, in this case 10, 20 and 60m resolutions will all be processed.

Please note that the activation of ESA-CCI data-package necessary for Sen2Cor plugin to generate products compatible with L2A Core products is not yet included in v0.8.

3.8 EOPF format convert plugin

In this updated version of TF, three new workflows are included, each built upon the conversion functions found in the eopf-cmp package v1.2.2.

The provided workflows are the following:

eopf_convert_to_zarr: to transform SAFE files into the Zarr format

eopf_convert_to_netcdf: to convert SAFE files into NetCDF format

eopf_convert_to_cog: to change SAFE files into Cloud Optimized Tiff format

Important considerations:

- The eopf-cmp v1.2.2's conversion functions are based on products released in 2022; any changes after this might lead to compatibility issues. Therefore the conversion of some product, even if supported, may fail.
- Currently, eopf-cmp v1.2.2 requires considerable memory during its processes, though this will be enhanced in forthcoming eopf versions. For instance, converting 2 GB images might necessitate over 8 GB of RAM. Therefore, the size of containers and the number of processes per worker (adjustable using the NPROCESSES variable in esa_tf/.env) should be tailored to the size of the products intended for conversion.

4. Logs and Outputs

All accessible logs of the Transformation Framework are printed to shell as standard output.

At the end of the processing, the TF will log the consumption information:

CPU usage

```
esa_tf-1.5.2-osf - esa_tf_platform.monitoring - order_id {order_id} – 18/05/2022 18:00:13.101 – INFO – total CPU Time: 1600.00 s
```

RAM usage

```
esa_tf-1.5.2-osf - esa_tf_platform.monitoring - order_id {order_id} – 18/05/2022 18:00:13.101 – INFO – peak RAM usage: 1.1 Gb
```

Disk usage

```
esa_tf-1.5.2-osf - esa_tf_platform.monitoring - order_id {order_id} – 18/05/2022 18:00:13.101 – INFO – peak disk usage: 2.1 Gb
```

Processing time and download time from the Data Source

```
esa_tf-1.5.2-osf - esa_tf_platform.monitoring - order_id {order_id} –18/05/2022 18:00:13.101 – INFO – wall time: 3600.00 s
```

4.1 NGINX access log

In the `LOGS` directory are stored logs file for the NGINX proxy. The `access.log` file contains every HTTP operation requested to the TF in a standard format.
Note: this is the only log file where you can monitor download attempts, as download operations never reach the OData API service.

4.2 Sen2Cor logs

The sen2cor plugin generates more specific logs in the `/root/sen2cor/2.9/log` path within the `esa_tf_esa_tf_worker_1` Docker container. In order to access this path, it is necessary to list all docker containers with the `docker container ls` command, and use the ID of that container to enter the container via command `docker exec -ti <ID> /bin/bash`.

Relevant information that can be inspected in the generated logs includes the check that the `Aerosol Type` option used is correctly recorded:

```
<message contentType="Text">Aerosol Type: Rural</message>
```

The following content appears when `Cirrus_Correction` is set to `True`:

```
<message contentType="Text">Cirrus Detection and Removal </message>
```

Information regarding other options is not reported; however, the logs also show some feedback on the download process such as:

```
<message contentType="Text">Trying to retrieve DEM from URL
http://srtm.csi.cgiar.org/wp-content/uploads/files/srtm_5x5/TIFF/ this may take some
time ...</message>
```

As the first download is successful, the following lines confirm the DEM has been downloaded (the tile name will vary, here `srtm_39_12.zip` is reported as an example):

```
<message contentType="Text">Zipfile downloaded: srtm_39_12.zip</message>
<message contentType="Text">DEM unpacked and moved: srtm_39_12.tif</message>
```

Alternatively, if the DEM file was already present in the folder, the following line will be printed:

```
<message contentType="Text">Dem exists: srtm_39_12.tif</message>
```

The expected output product will be a Sentinel-2 L2A in zip format. The relevant data will be contained in a SAFE folder within the zip. The output will follow standard naming convention, hence the filename of the output file is unchanged w.r.t. the input reference one, except for the PDGS Processing Baseline number (Nxxyy) and the second date (at the end of the filename) where the <Product Discriminator> field is substituted by the creation date. Please note that in the Creation Date is here the date of creation of the output file on behalf of sen2cor plugin (at the beginning of the process), which differs from the submission date.

5. Transformation Framework Usage

This section may act as a quick manual for users of the Transformation Framework, gathering short examples of commands usage.

5.1 Request a new TransformationOrder

In order to request a new TransformationOrder, it is possible to use the command `curl`, passing through a JSON string with the full desired workflow options. The following is a sample request:

```
curl -v -d '{"WorkflowId": "sen2cor_l1c_l2a", "InputProductReference": {"Reference":
"S2B_MSIL1C_20211109T110159_N0301_R094_T29QQB_20211109T114303.zip", "DataSourceName":
"scihub"}, "WorkflowOptions": {"Aerosol_Type": "RURAL", "Mid_Latitude": "AUTO",
"Ozone_Content": 0, "Cirrus_Correction": true, "DEM_Terrain_Correction": true, "Resolution":
10}}' -H "Content-Type: application/json" http://localhost:8080/TransformationOrders | jq
```

Please note that, as in the example above, the Content-Type and address must respectively always remain `application/json` and `http://<IP>:<PORT>/TransformationOrders`.

When the TF receives a TransformationOrder request:

- if TransformationOrder is not jet in the list of the submitted orders, then the TF will submit the new order.
- if TransformationOrder is already in the list of the submitted orders, then:
  - if it is in *progress* or if it is *completed* and the output product is available, the TF does not resubmit the order
  - if it is *completed*, but the output product has been deleted, the TF resubmit the order
  - if it is *failed*, the TF resubmit the order

Once the TransformationOrder is submitted, if the DataSource parameter is defined, the TF will try to download the product from the selected data source. If the DataSource parameter is not explicitly defined, the TF will try to download the product from the hubs defined in hubs_credentials.yaml, starting with the first one. If an error occurs, it will try to download data from the next source, until the data is downloaded correctly or the sources run out.

If the user does not define one of the available workflow options in the request, the default value corresponding to this option is used in the order.

If the request is successful, the following information on the transformation order is returned:

```
{
  "Id": "9ad5f1264795f663f7d7a36daedf0635",
  "InputProductReference": {
    "Reference": "S2B_MSIL1C_20211109T110159_N0301_R094_T29QQB_20211109T114303.zip",
    "DataSourceName": "scihub",
  },
  "WorkflowOptions": {
    "Aerosol_Type": "RURAL",
    "Mid_Latitude": "AUTO",
    "Ozone_Content": 0,
    "Cirrus_Correction": true,
    "DEM_Terrain_Correction": true,
    "Resolution": 10,
  },
```

```
  "WorkflowId": "sen2cor_l1c_l2a",
  "WorkflowName": "Sen2Cor_L1C_L2A",
  "SubmissionDate": "2021-11-25T15:11:05",
  "Status": "in_progress"

}
```


5.2 Monitor the status of a transformation order

Once the order has been submitted, its status can be monitored with the following URL, where the order ID is passed through as argument:

```
http://<IP>:<PORT>/TransformationOrders('9ad5f1264795f663f7d7a36daedf0635')
```

The corresponding response will then be:


```
{
  "Id": "9ad5f1264795f663f7d7a36daedf0635",
  "InputProductReference": {
    "Reference": "S2B_MSIL1C_20211109T110159_N0301_R094_T29QQB_20211109T114303.zip",
    "DataSourceName": "scihub",
  },
  "WorkflowOptions": {
    "Aerosol_Type": "RURAL",
    "Mid_Latitude": "AUTO",
    "Ozone_Content": 0,
    "Cirrus_Correction": true,
    "DEM_Terrain_Correction": true,
    "Resolution": 10
  },
  "WorkflowId": "sen2cor_l1c_l2a",
  "WorkflowName": "Sen2Cor_L1C_L2A",
  "SubmissionDate": "2021-11-25T15:11:05",
  "Status": "in_progress"

}
```


Once the Order is completed, the response becomes:


```
{
  "Id": "9ad5f1264795f663f7d7a36daedf0635",
  "InputProductReference": {
    "Reference": "S2B_MSIL1C_20191025T085939_N0208_R007_T37VCC_20191025T112031.zip",
    "DataSourceName": "scihub",
  },
  "WorkflowOptions": {
    "Aerosol_Type": "RURAL",
    "Mid_Latitude": "AUTO",
    "Ozone_Content": 0,
    "Cirrus_Correction": true,
    "DEM_Terrain_Correction": true,
    "Resolution": 10
  },
  "WorkflowId": "sen2cor_l1c_l2a",
  "WorkflowName": "Sen2Cor_L1C_L2A",
  "SubmissionDate": "2021-11-25T15:11:05",
  "Status": "completed",
```

```
  "CompletedDate": "2021-11-25T15:12:05",
  "OutputProductReference": [
        {
            "Reference": "S2B_MSIL2A_20191025T085939_N9999_R007_T37VCC_20211125T151206.zip",
            "DownloadURI":
"http://<IP>:<PORT>/download/9ad5f1264795f663f7d7a36daedf0635/S2B_MSIL2A_20191025T085939_N99
99_R007_T37VCC_20211214T093626.zip"
        }
    ]
}
```


5.3 Obtain a list of transformation orders

The list of the transformation orders submitted by a user can be retrieved via the following URL:

`http://<IP>:<PORT>/TransformationOrders`

It is also possible to filter accessible orders by status type. In this example, the URL queries the list of completed orders:

curl "http://localhost:8080/TransformationOrders?\$filter=Status%20eq%20'completed'" | jq

The user with manager profile can request also the list of the transformation submitted by all users:

`http://<IP>:<PORT>/admin/TransformationOrders`

5.3.1 Transformation orders responses

A response obtained with the latter query is for instance the following:


```
{
  "@odata.id": "http://<IP>:<PORT>/TransformationOrders('9ad5f1264795f663f7d7a36daedf0635')",
  "Id": "9ad5f1264795f663f7d7a36daedf0635",
  "InputProductReference": {
    "Reference": "S2B_MSIL1C_20191025T085939_N0208_R007_T37VCC_20191025T112031.zip",
    "DataSourceName": "scihub",
    "ContentDate": {
      "Start": "2019-10-25T08:59:39.922Z",
      "End": "2019-10-25T11:20:31.922Z"
    }
  },
  "WorkflowOptions": {
    "Aerosol_Type": "RURAL",
    "Mid_Latitude": "SUMMER",
    "Ozone_Content": 331,
    "Cirrus_Correction": false,
    "DEM_Terrain_Correction": true,
    "Resolution": 10
  },
  "WorkflowId": "sen2cor_l1c_l2a",
  "WorkflowName": "Sen2Cor_L1C_L2A",
  "SubmissionDate": "2021-12-14T10:12:40",
  "Status": "completed",
```

```
  "CompletedDate": "2021-11-25T15:12:05",
  "OutputProductReference": [
        {
            "Reference": "S2B_MSIL2A_20191025T085939_N9999_R007_T37VCC_20211125T151206.zip",
            "DownloadURI":
"http://<IP>:<PORT>/download/9ad5f1264795f663f7d7a36daedf0635/S2B_MSIL2A_20191025T085939_N99
99_R007_T37VCC_20211214T093626.zip"
        }
    ]
}
```


Conversely, if no transformation order is found, the response will be:


```
{
  "detail": "Transformation order 9ad5f1264795f663f7d7a36daedf0635 not found"
}
```