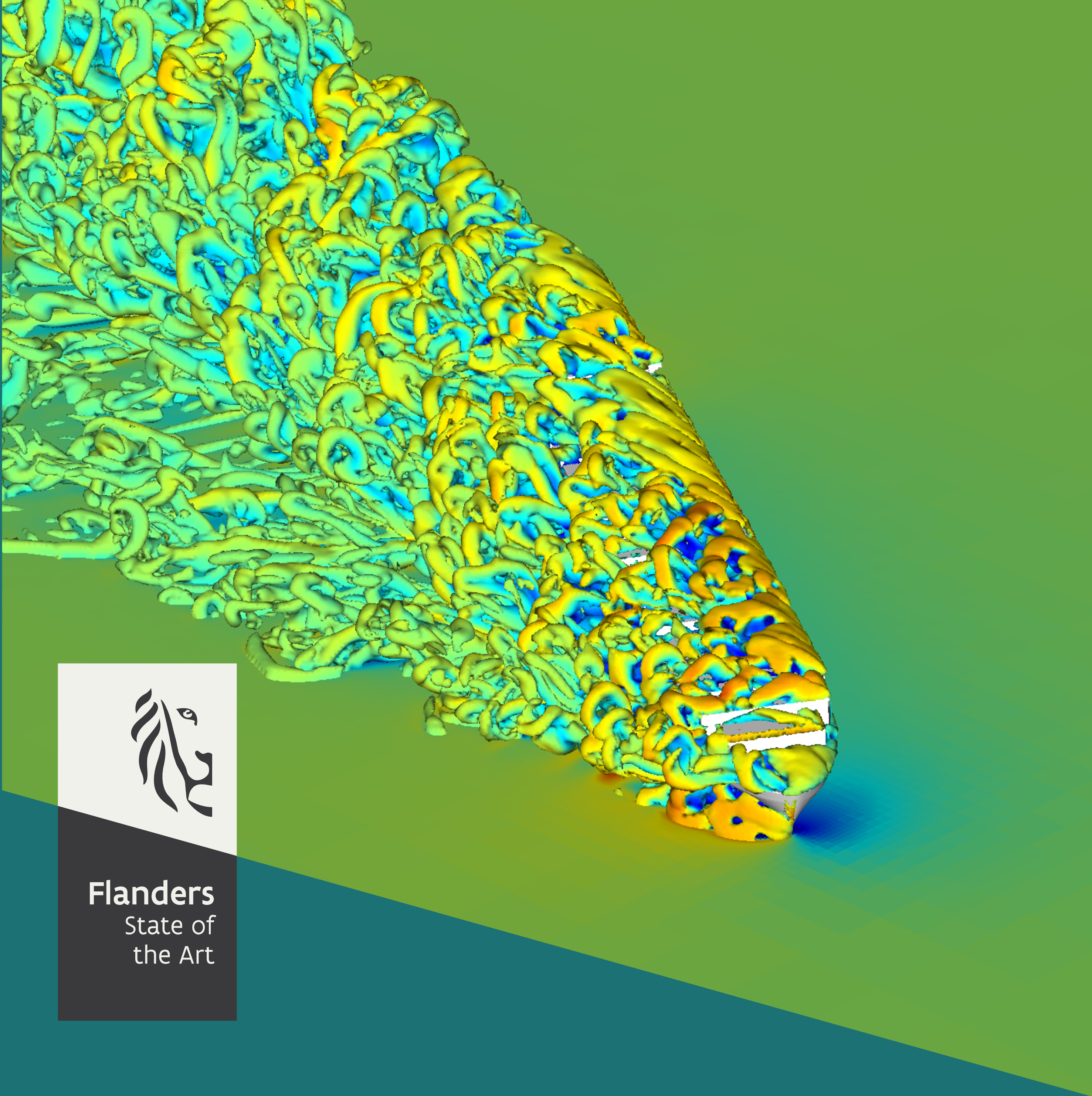


PA032_3
FH Reports

Beheer en Onderhoud CFD modellen

Updates to the CFD CAD model database

DEPARTMENT
MOBILITY & PUBLIC WORKS

www.flandershydraulics.be

Beheer en Onderhoud CFD modellen

Updates to the CFD CAD model database

Van Hoydonck, W.; Lopez Castano, S.; Panahi, S.

Cover figure © The Government of Flanders, Department of Mobility and Public Works, Flanders Hydraulics

Legal notice

Copyright and citation

© The Government of Flanders, Department of Mobility and Public Works, Flanders Hydraulics, 2025

D/2025/3241/141

This publication should be cited as follows:

Van Hoydonck, W.; Lopez Castano, S.; Panahi, S. (2025). Beheer en Onderhoud CFD modellen: Updates to the CFD CAD model database. Version 2.0. FH Reports, PA032_3. Flanders Hydraulics: Antwerp

Reproduction of and reference to this publication is authorised provided the source is acknowledged correctly.

Document identification

Customer:	WL	Ref.:	WL2025RPA032_3
Keywords (3-5):	CFD, CAD, database, Confluence wiki, Git, Bitbucket, Python		
Knowledge domains:	Harbours and waterways > Manoeuvring behaviour > Open Water > Numerical calculations		
Text (p.):	9	Appendices (p.):	0
Confidential:	No	☒ Available online	

Author(s):	Van Hoydonck, W.

Control

	Name	Signature
Revisor(s):	Lopez Castano, S.; Panahi, S.	Getekend door:LOPEZ CASTANO Santia Getekend op:2025-06-02 14:33:43 +02:0 Reden:Ik keur dit document goed Lopez Castano Santiago Vlaamse overheid — Getekend door:Saeid PANAHI (Signature Getekend op:2025-06-04 08:41:08 +02:0 Reden:I approve this document Panahi Saeid
Project leader:	Van Hoydonck, W.	Getekend door:Wim Van Hoydonck (Sign Getekend op:2025-06-02 13:56:15 +02:0 Reden:Ik keur dit document goed Van Hoydonck Wim Vlaamse overheid

Approval

Head of division:	Bellafkih, A.	Getekend door:Abdelkarim Bellafkih (Sign Getekend op:2025-06-02 13:57:31 +02:0 Reden:Ik keur dit document goed Bellafkih Abdelkarim Vlaamse overheid

Abstract

The objective of this report is to document recent modifications to the Computational Fluid Dynamics (CFD) Computer Aided Design (CAD) model database and its Confluence Wiki front-end. Due to the migration of the on-premises Confluence wiki installation to a cloud-hosted Confluence solution (with a newer version), the wiki front-end required adaptations. Due to increased security measures included in the cloud-based Confluence wiki, the authentication to adapt pages via the REST API is different from the default authentication to edit pages manually. In addition to these changes, the subversion back-end was migrated to Bitbucket (git) to enable external access to the repository. The Python script that was used to create the Confluence wiki pages, has been updated with a Graphical User Interface (GUI) as well, to improve the user experience.

Contents

List of Figures

Nomenclature

Abbreviations

CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
FH	Flanders Hydraulics
GUI	Graphical User Interface

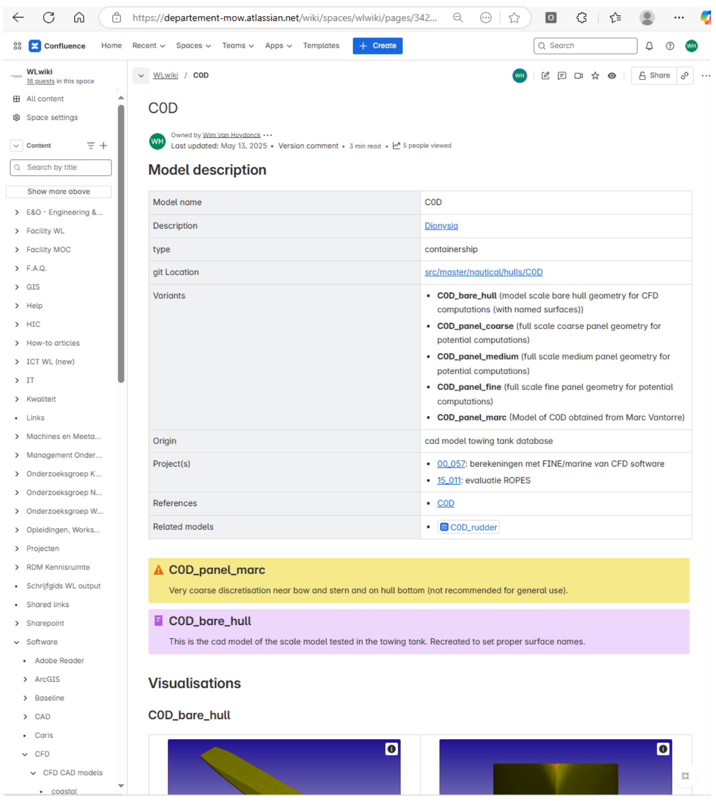
1 Introduction

1.1 Background

In Van Hoydonck *et al.* (2020), the rationale for and initial implementation of the CFD CAD model database have been documented. The database is used to document model specific information in a clear and structured way. The system consists of a back-end (version controlled database) where for each CAD model, all input files, intermediate files and final files are stored together with visualisations of the CAD models and an `info.xml` file that contains (textual) information about the model such as:

- the origin of the data;
- a description of variants of the model with possible peculiarities;
- related models;
- projects in which the model has been used;
- relevant references (such as external websites, or reports);
- and a list of files with metadata and a description.

The contents of these `info.xml` files together with the visualisations are used by a Python script to create (or update) a wiki page that displays all relevant data in a visual manner. An example of the top part of one such page (for ship C0D) is shown in Fig. 1. This report documents in Chapter 2 the required modifications to the Python script to create the wiki pages and the front-end of said Python script.



C0D

Owned by Wim Van Hoydonck
Last updated: May 13, 2025 • Version comment • 3 min read • 5 people viewed

Model description

Model name	C0D
Description	Dionysia
type	containership
git Location	src/master/nautical/hulls/C0D
Variants	• **C0D_bare_hull** (model scale bare hull geometry for CFD computations (with named surfaces)) • **C0D_panel_coarse** (full scale coarse panel geometry for potential computations) • **C0D_panel_medium** (full scale medium panel geometry for potential computations) • **C0D_panel_fine** (full scale fine panel geometry for potential computations) • **C0D_panel_marc** (Model of C0D obtained from Marc Vantorre)
Origin	cad model towing tank database
Project(s)	• 00_057: berekeningen met FINE/marine van CFD software • 15_011: evaluatie ROPES
References	• C0D
Related models	• C0D_rudder

C0D_panel_marc

Very coarse discretisation near bow and stern and on hull bottom (not recommended for general use).

C0D_bare_hull

This is the cad model of the scale model tested in the towing tank. Recreated to set proper surface names.

Visualisations

C0D_bare_hull

Figure 1 – Example CAD model page on the Confluence wiki of FH

1.2 Database components, locations and access

1.2.1 Database components and locations

The back-end of the CFD CAD model database is located on Bitbucket:

`https://bitbucket.org/departement-mow/cfd_cad_models/src/master/`.

The front-end is located on Confluence:

`https://departement-mow.atlassian.net/wiki/spaces/wlwiki/pages/342166393/CFD+CAD+models`.

The Python script (`confluence_wiki.py`) to validate existing or new `info.xml` files and to update existing wiki pages and create new pages is located in the top-level directory of the Bitbucket repository:

`https://bitbucket.org/departement-mow/cfd_cad_models/src/master/confluence_wiki.py`.

1.2.2 Access

In order to access the Bitbucket repository, a username and password are required. These can be obtained from `WL-IT` or the author of this report.

For access to the Confluence wiki, an additional username and password are required, that can also be obtained from `WL-IT`.

With the above credentials, the wiki can be accessed and CAD models can be utilized. If one wants to add new or updated models to the repository and create the wiki front-end page for it, additional authentication is required to enable interaction with the Confluence server through its REST API. The required steps to create one such API token is described below.

Log in with your normal Confluence credentials and in the top-right corner, open the account drop-down menu and select the *Manage Account* hyperlink (Fig. 2). Select the *Security* category at the top of the page and press *Create and manage API tokens* (Fig. 3).

Despite the warning at the top of the next page (Fig. 4) that API tokens without scope are deprecated, a *token without scope* will be created[1] (see Fig. 5). A name and end date (at most 365 days in the future) must be specified. If valid values are given for both criteria, the API token can be created. The token should be saved in a secure location, such as in a password manager on your computer[2]. Due to the limited validity of the API tokens, a new token has to be created yearly.

When in the next chapter the GUI of the Python script is discussed (section 2.3), an explanation is given on how the token can be added to a user and stored on the computer.

[1]Creating an API token with scope should work as well (which is recommended by Atlassian), but it is unclear which of the 86 available scopes must be selected to make the script work without errors. As a maximum of 50 scopes can be selected for an API token (and selecting all *Classic* scopes seems to be insufficient), figuring this out will be left as a future task.

[2]Printed out and stored in a vault in an underground cave (known only to you) located in an area on earth where earthquakes rarely occur is another possibility.

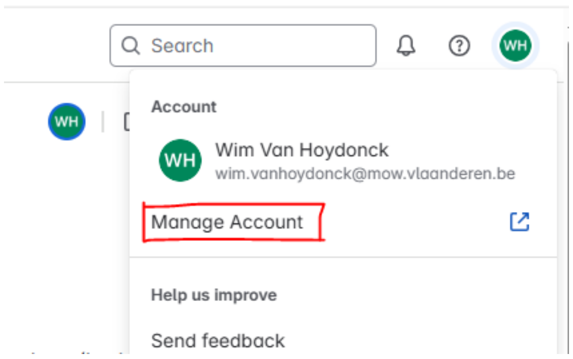
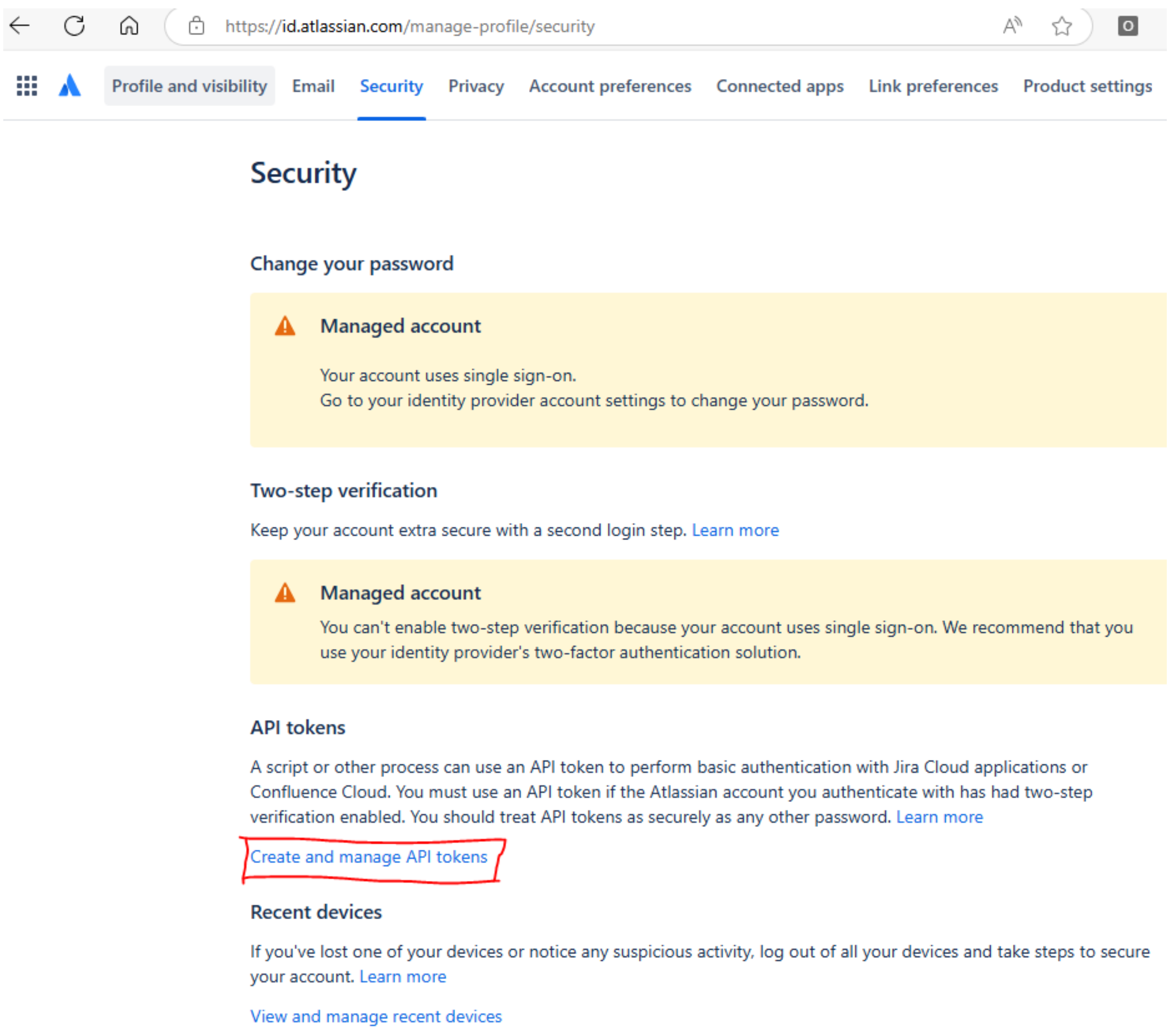


Figure 2 – Confluence: manage account.



Security

Change your password

Managed account

Your account uses single sign-on.
Go to your identity provider account settings to change your password.

Two-step verification

Keep your account extra secure with a second login step. Learn more

Managed account

You can't enable two-step verification because your account uses single sign-on. We recommend that you use your identity provider's two-factor authentication solution.

API tokens

A script or other process can use an API token to perform basic authentication with Jira Cloud applications or Confluence Cloud. You must use an API token if the Atlassian account you authenticate with has had two-step verification enabled. You should treat API tokens as securely as any other password. Learn more

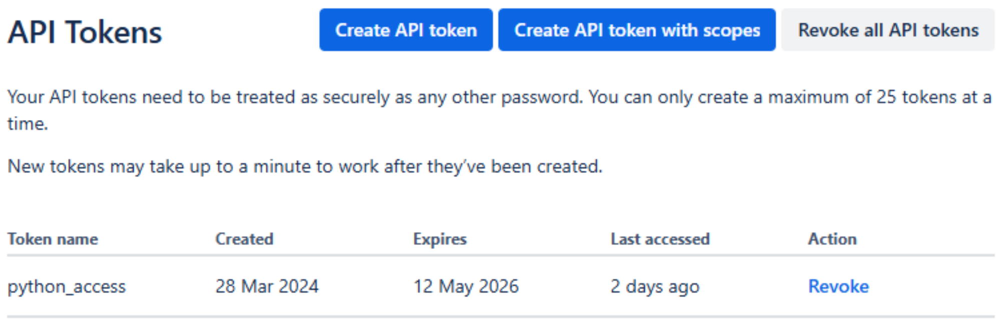
Create and manage API tokens

Recent devices

If you've lost one of your devices or notice any suspicious activity, log out of all your devices and take steps to secure your account. Learn more

View and manage recent devices

Figure 3 – Confluence: create and manage API tokens.

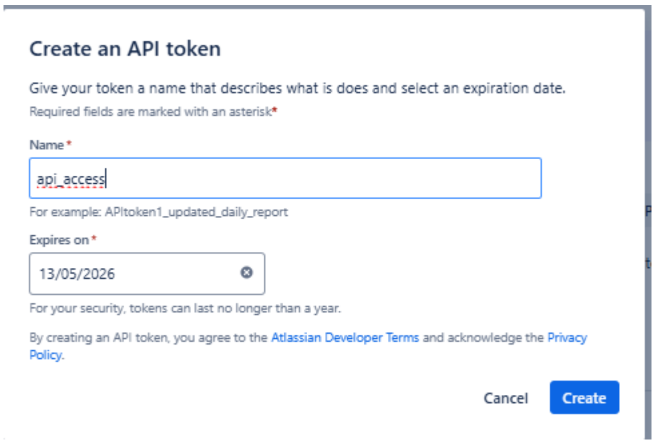

API Tokens

Your API tokens need to be treated as securely as any other password. You can only create a maximum of 25 tokens at a time.

New tokens may take up to a minute to work after they've been created.

Token name	Created	Expires	Last accessed	Action
python_access	28 Mar 2024	12 May 2026	2 days ago	Revoke

Figure 4 – Confluence: create API tokens.


Create an API token

Give your token a name that describes what is does and select an expiration date.
Required fields are marked with an asterisk*

Name*
api_access
For example: APItoken1_updated_daily_report

Expires on*
13/05/2026
For your security, tokens can last no longer than a year.

By creating an API token, you agree to the Atlassian Developer Terms and acknowledge the Privacy Policy.

Figure 5 – Confluence: create a new API token without scope.

2 Python script modifications

Modifications to the Python script to update existing wiki pages or create new ones are discussed below. First, the Confluence REST API access is discussed. This is followed by a discussion on the necessary changes to the page content (such as hyperlinks) and finally, the addition of a graphical user interface is shortly discussed.

2.1 Confluence connection via REST API

As explained in the previous chapter, the default username and password used to access and edit Confluence wiki pages are not sufficient to access its REST API. For that, a separate token is required. Using this authentication, the implementation from 2020 (using the third-party *atlassian-python-api* package) did not fully work. Amongst others, updating an existing page with new content or creating a new page from scratch did not work. Extracting page content of existing pages did work. An attempt to debug the inner workings of this library to find the culprit was not successful.

With the help of an IT subcontractor of the Flemish government, a workaround was created where the intermediate library was not used and calls to create new pages or update existing ones were made directly using the *requests* library (which is also used in *atlassian-python-api*). With a bit of work, this proved successful. A small class *wiki_page* was created that implements the following methods to execute common tasks:

get_space_id return the numeric id of a space given its name;

get_page return the page content of the supplied page name;

get_parent_page_id return the id of the parent of the supplied page;

get_page_id return the id of the supplied page;

create_page create a new page with a given name and content as child of the supplied parent page;

get_page_content return page body of the supplied page;

update_page replace the existing content of the supplied page with the new content, or append the new content to the existing content;

set_page_label set a label on a page;

get_child_pages return a list of all child pages of a supplied page.

With the exception of one function (*set_page_label*), all of the above methods use the Confluence Cloud REST API V2.

2.2 HTML page changes

The HTML body of the wiki pages is created using the same functions as initially implemented in 2020 with some small modifications due to changes in the Confluence wiki software. The changes are related to how references to external (image) content should be created and the method to add *warning*, *info*, *error* and other panels on a wiki page.

2.2.1 Referencing external images as hyperlinks

In the original implementation, referencing an external image in a wiki page was using the standard HTML `<img />` tag with attributes `class`, `width`, `src` and `data-image-src`, as such:

```
<img class="confluence-embedded-image confluence-external-resource" width="500"
  src="{img_url}{f}" data-image-src="{img_url}{f}" />
```

where `img_url` and `f` combined point to the location of the image in the version control system.

In the new implementation, the HTML `<img />` tag is replaced with the following XHTML code:

```
<ac:image ac:align="center" ac:width="300" ac:alt="{f}" ac:src="{img_url}{f}">
  <ri:url ri:value="{img_url}{f}" />
</ac:image>
```

where again, `img_url` and `f` combined point to the location of the image. The above method is not documented by Atlassian and the online wiki editor does not even contain the ability to directly add a hyperlink to an external image. Instead, the following markdown code can be used[3] in the editor to create a hyperlink to an external image:

```
![image explanation](https://url/to/image)
```

By subsequently inspecting the body of a page where such a link was added, the XHTML alternative using the combination of `<ac:image>` and `<ri:url >` was discovered.

The change from Subversion to git as revision control system means that all URLs to images displayed in the wiki must be adapted to point to the location of the Bitbucket repository. To display a file located a Bitbucket repository as-is without the web interface around, the URL has to be changed. As an example, one of the visualisations of ship `COD` shown at the bottom in the screenshot in Fig. 1, has the following URL in Bitbucket:

```
https://bitbucket.org/departement-mow/cfd_cad_models/src/master/nautical/hulls/
COD/vis/COD_bare_hull_ISO.png
```

However, in a browser, this URL does not show the image alone, but as part of the Bitbucket browser interface. The URL must be modified to point to `raw/HEAD` instead of the `src/master`:

```
https://bitbucket.org/departement-mow/cfd_cad_models/raw/HEAD/nautical/hulls/
COD/vis/COD_bare_hull_ISO.png
```

2.2.2 Warning, info, note and tip panels

In the original implementation of 2020, formatted text could be created using a number of predefined elements (*info*, *warning*, *tip*, …) to prominently show notes related to a CAD model or one of its variants. After the migration to the online Confluence wiki, these stopped working as well and had to be replaced with new code. The remark template shown below requires three parameters (`icon`, `iconId` and `bgColor`) to associate it with one of the five different panel types in addition to a `body` and optional `title` (if the remark is related to a specific model variant).

```
remark_template = """
    <ac:structured-macro ac:name="panel" ac:schema-version="1">
      <ac:parameter ac:name="panelIcon">{icon}</ac:parameter>
      <ac:parameter ac:name="panelIconId">{iconId}</ac:parameter>
      <ac:parameter ac:name="bgColor">{bgColor}</ac:parameter>
      <ac:rich-text-body><h2>{title}</h2><p>{body}</p></ac:rich-text-body>
```

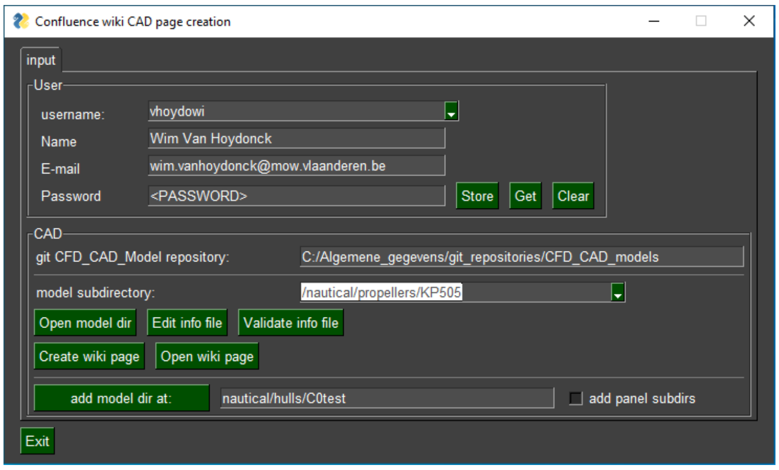
[3] As discussed in one of the comments to the following Jira issue: `https://jira.atlassian.com/browse/CONFCLOUD-65749`.

```
    </ac:structured-macro>
    """
```

2.3 Graphical User Interface

To improve the user experience of creating new wiki pages based on added or updated ?? models, a graphical user interface (GUI) was added to Python script. In the original implementation of 2020, the Python script (`create_wlwiki_page.py`) contained a `main` function that had to be adapted manually to create a wiki page for a specific CAD model. A new Python script (`confluence_wiki.py`) was developed from scratch instead of adapting the original Python script. The majority of the code used to construct the different parts of the wiki pages was copied into the new script without requiring adaptations. A YAML file in the base directory of the CAD model database (`confluence_users.yaml`) contains users for which the password (API token) to access Confluence via the REST API can be stored. Note that a new user needs to add him- or herself to this file and commit it. The structure for user *johndoe* in that file can be used as template. The API token of a user can be stored on and retrieved from the computer by the GUI (using the `keyring` Python package). Note that this method only stores the API token on the local computer of the user. If access from multiple computers is required, either the existing API token has to be copied, or a new token has to be created for each computer from which access is required.

A screenshot of the GUI is shown in Fig. 6. Based on the user that is currently logged in, the name and email field are pre-filled with the data from `confluence_users.yaml`. If, in addition, the checkout field in that file points to a valid location on the computer of the user, the git repository location is filled in as well. The drop-down menu *model subdirectory* contains a list with all subdirectories below the git checkout directory where CAD models may be located. For *nautical* models (hulls, rudders, propellers, …), the list contains all directories two levels below the *nautical* directory, while for the *coastal* and *hydraulic* directories, it contains all directories at one level lower.


Confluence wiki CAD page creation

User
username: vhoydowi
Name: Wim Van Hoydonck
E-mail: wim.vanhoydonck@mow.vlaanderen.be
Password: <PASSWORD>

CAD
git CFD_CAD_Model repository: C:/Algemene_gegevens/git_repositories/CFD_CAD_models
model subdirectory: /nautical/propellers/KP505
add model dir at: nautical/hulls/C0test
add panel subdirs

Figure 6 – GUI of `confluence_wiki.py` to create the wiki pages for a specific CAD model.

When a CAD model directory is selected, the following tasks can be executed:

- the directory can be opened in Windows Explorer;
- the `info.xml` file in the directory can be edited in a text editor defined using the `EDITOR` environment variable;
- the `info.xml` file can be validated;
- a wiki page can be created from the data in the currently selected model subdirectory.

In addition to these tasks related to an existing CAD model in the git repository, a relative path can be added in the bottom text field to add an empty model directory with the majority of the subdirectories present. A `info.xml` file is added to the main directory as a starting point for the user. When adding a new directory structure this way, the new model subdirectory is added to the list of CAD model directories in the drop-down menu with all model subdirectories. Functionality to open an existing or newly created wiki page in a browser was added as well.

3 Conclusions and Discussion

In 2020, a report was written to document the implementation of a database to store CAD models used in CFD computations. Changing requirements related to the back-end, migration of the front-end to a cloud-based wiki installation and a desire to simplify the user experience are the main reasons to write the current report.

The Subversion repository (back-end) was migrated to Bitbucket in 2024 to allow external access to the CAD models. In that same year, the on-premises Confluence wiki was migrated to a cloud-hosted Confluence wiki installation. The latter migration was not fully backward compatible, which required modifications to the Python script to create the wiki pages. The Python script to create the CAD model wiki pages was augmented with a GUI to speed up the process of creating the wiki pages.

At the time of writing this report, the database contains 28 CAD models of ship hulls, 11 ship rudder models, three propeller models and three other models in the nautical section. The hydraulic section contains at the moment 7 models related to hydraulic structures. As compared to the state of the database in 2020, there are now more than twice the number of models in the database.

At the moment, the files stored in the subdirectories of a CAD model are only related to the geometry of the model. There is however no fundamental restriction to the type or list of files stored. For example, for a small number of nautical *hull* models[4], a file (*HXP_ref_dict.csv*) is included with mesh generation settings for HEXPRESS that can be used to generate a base mesh for that specific model. A dictionary with settings related to OpenFOAM meshers such as SnappyHexMesh or cfMesh could be stored there as well.

[4]At the time of writing this report, a Hexpress dictionary file is present for hull models *M2014A* (SHINING inland hull), *T0Z* (KVLCC2) and *C04* (KCS).

DEPARTMENT **MOBILITY & PUBLIC WORKS**
Flanders Hydraulics

Berchemlei 115, 2140 Antwerp
T +32 (0)3 224 60 35
F +32 (0)3 224 60 36
waterbouwkundiglabo@vlaanderen.be
www.flandershydraulics.be