CADMATIC Web API

User Guide | 2024T3

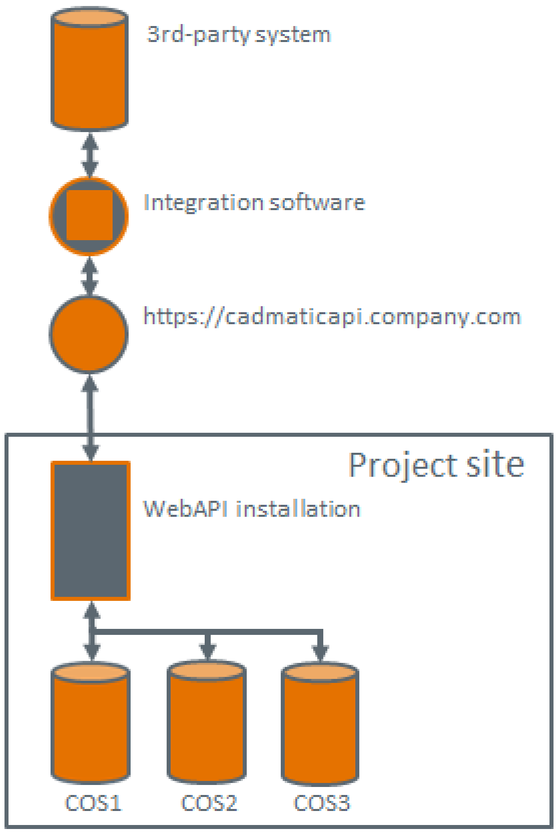
Cadmatic
Linnankatu 52
20100 Turku
Finland
Tel. +358 2 412 4500
www.cadmatic.com

Contents

1. CADMATIC Web API

The CADMATIC Web API is a REST API that provides interfaces for seamless transferring of 3D CAD project data between a 3rd party system (ERP, PLM) and CADMATIC Plant/Outfitting (in the future, also CADMATIC Hull). Using a Web API for the data transfer makes the data more coherent across the different systems and reduces repetitive manual work, for example, by retrieving document publication files automatically. The Web API can be accessed with any software client that can make simple HTTP requests over the network, or it can be used manually via a web browser.

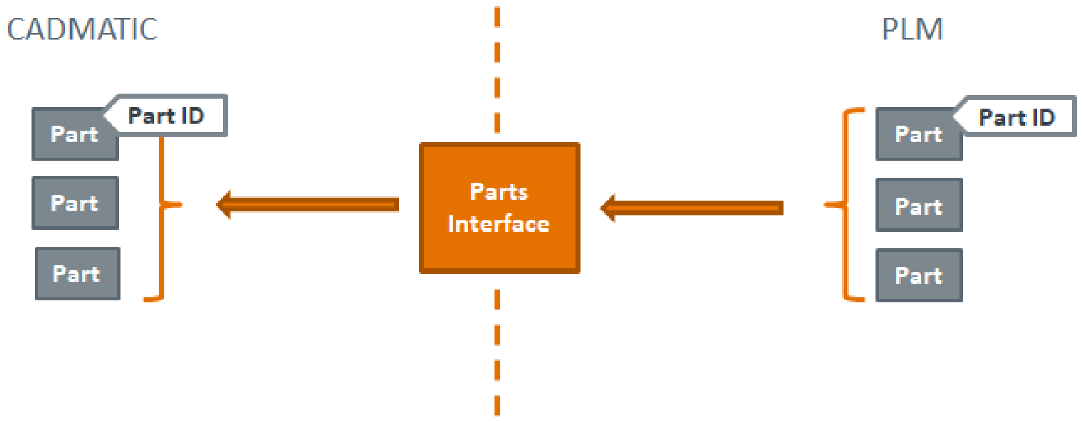
The Web API works independently of the Cadmatic design applications: it only interacts with the Common Object Storage (COS) databases that have been added to its configuration. You can install the Web API to any site that has a COS installation; you can connect multiple COS servers from different projects and external databases to the same Web API.



The Web API is run as a Windows service, and there can be several service instances and application versions running simultaneously. Starting the service attempts to open a session to the COS servers that have been configured for that service instance, and after the service has established the connection, other systems can start accessing the data via the API. The service runs an embedded web server that receives incoming requests and returns the requested information to the requesting party. The API does not store the state between separate requests, which simplifies communication across different integration solutions as all communication can happen in parallel.

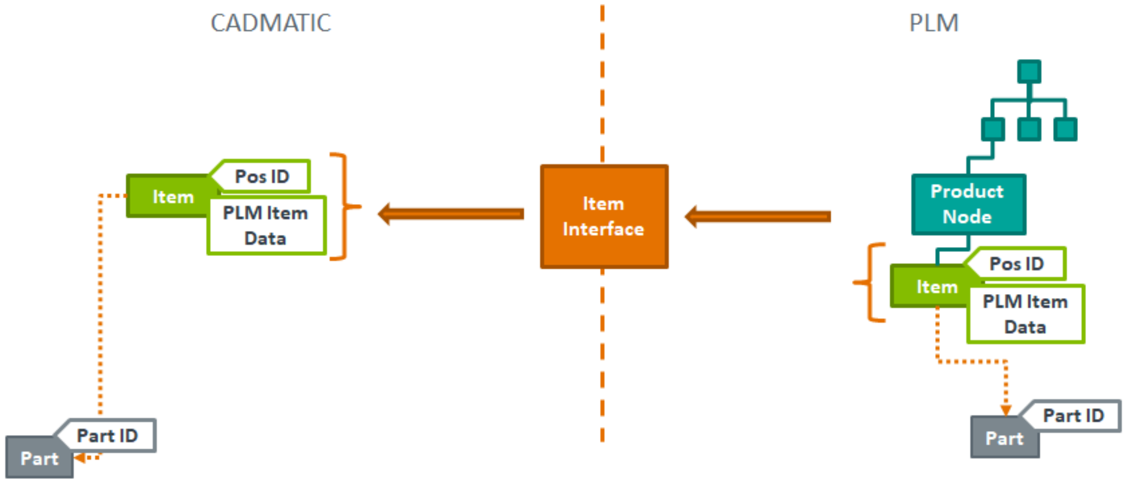
The Web API has interfaces for the following:

- Authorization
- Changes
    - Get the objects that have changed in a database after a given point. This is intended for a connecting software that informs another database when COS objects have been created or updated. The connecting software does not need to remember the last state in COS, it can just ask for a list of objects that have changed since the last time it asked.
- 3D exports and imports, in various formats.
- Catalog parts & part sizes
    - You can update part attribute data, such as the material number, from another system to Cadmatic applications.



- COS objects
    - Read the attributes and properties of any COS objects using their OID.
    - Read references to COS objects.
- Document publication files
    - When the design documents are ready, you can fetch the final documents to the project's repository using CADMATIC Web API.
- Objects with a position ID and attributes

- You can create, for example, equipment, valves, instruments and their attribute data from an ERP or PLM before starting the actual design work in Cadmatic applications. The designer can then use the ready-made objects in the 3D model or P&I diagram.
- Users can also link the object from, for example, PLM or ERP to the instance of the object in Cadmatic applications.

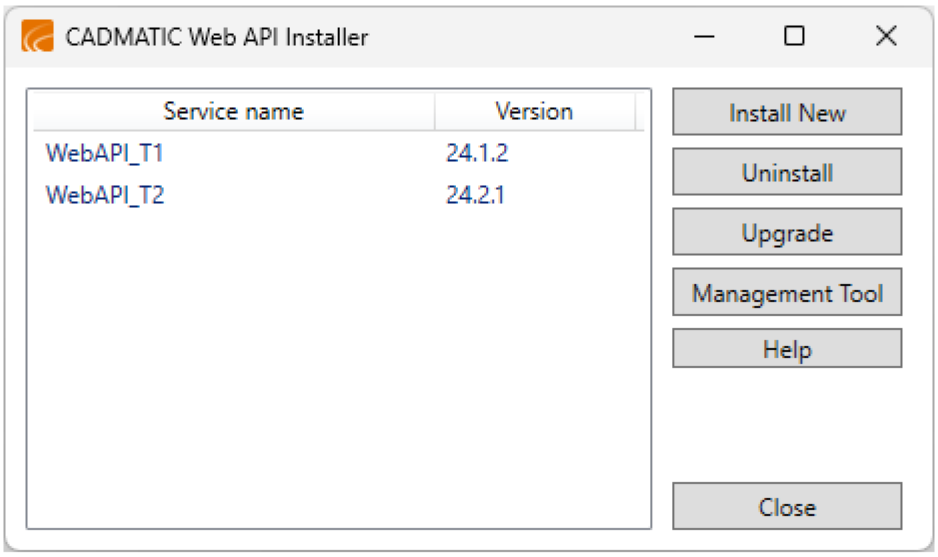


- Equipment GDL, Structural GDL, Parametric models
  - Read, create, modify, or import GDL for equipment and structurals.
  - Read and modify attributes of parametric models for equipment and structural GDLs.
- EDM object data
  - Bring external data management (EDM) object data to the CADMATIC 3D model or the CADMATIC piping and instrumentation (P&I) diagram.
- EPD object data
  - Bring external P&ID data to the CADMATIC 3D model.
- Groups
  - Get a list of groups or the properties of a group.
- Plant Modeller
  - Use model queries to get information from the 3D model objects of a Plant Modeller project.
- Servers
  - Get information from COS servers.
- Specifications

  - List piping, cable tray, and ducting specifications, along with the parts belonging to each specification.
- Status
  - Get the status of connectors.
- Systems and pipelines
  - List all systems and pipelines

2. CADMATIC Web API Installer

You can use the **CADMATIC Web API Installer** dialog to install, upgrade, and uninstall CADMATIC Web API services, and to access the management tools of the Web API.



Service name	Version
WebAPI_T1	24.1.2
WebAPI_T2	24.2.1

To access this dialog, do one of the following:

- Select **Start > CADMATIC > Web API Installer**.
- Browse to the *%PMS_RUNDIR%/webapi/WebApiInstaller* folder of your CADMATIC Plant/Outfitting installation, and run the *WebApiInstaller.exe* file as an administrator.

Note: %PMS_RUNDIR% refers to the path *C:\Cadmatic\cxxx\pmsxxx0.nt* (Plant) or *C:\Cadmatic\coxxx\pmsxxx0.nt* (Marine) where xxx is the version number of the CADMATIC release.

2.1. Installing a service

You can install multiple CADMATIC Web API services, and different versions of the services, to the same environment.

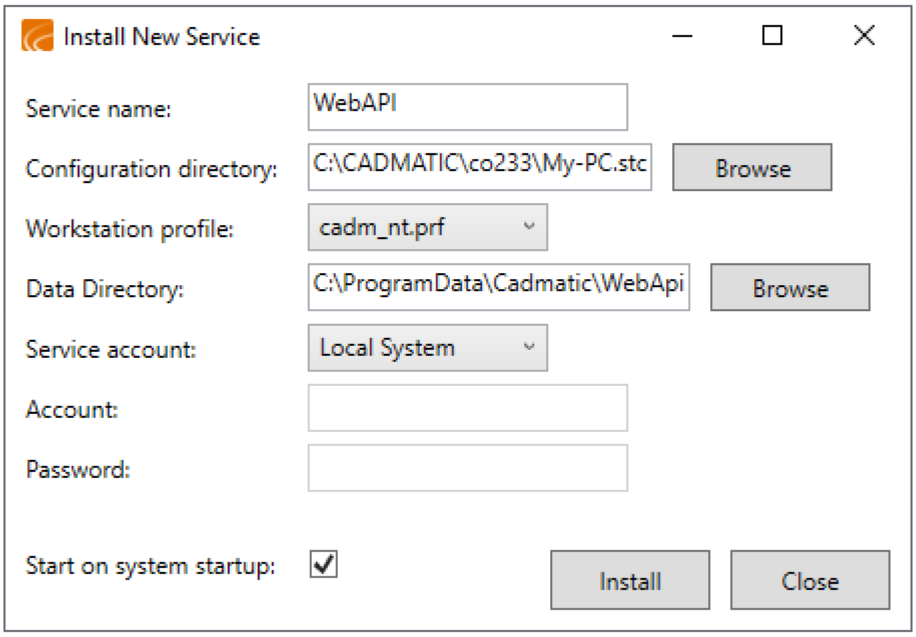
In a multi-site scenario, the Web API can be installed to an online replica site. Some of the endpoints request the ownership of objects directly from COS.

Prerequisites

- You have administrator privileges on the target computer.

Do the following:

1. Open the **CADMATIC Web API Installer** dialog.
2. Click **Install New**. The **Install New Service** dialog opens.



Install New Service

Service name:	WebAPI
Configuration directory:	C:\CADMATIC\co233\My-PC.stc
Workstation profile:	cadm_nt.prf
Data Directory:	C:\ProgramData\Cadmatic\WebApi
Service account:	Local System
Account:	
Password:	
Start on system startup:	✓

3. Define the following settings:
   - **Service name** – Enter a unique name for the service. The name cannot contain spaces.
   - **Configuration directory** – Select the CADMATIC Plant/Outfitting configuration folder.

   Tip: The configuration folder has the ".stc" suffix in its name (<*workstationname*>*.stc*), and it contains the workstation profiles in the *wsprofile* subfolder.

   - **Workstation profile** – Select the workstation profile that the service is to use.
   - **Data Directory** – The service stores its data in *%ProgramData%\CADMATIC\WebApi\<service name>* by default; click **Browse** if you want to change this location.

   Important: The data folder must be empty and located on a local drive: you cannot use a network drive for this. The service account must have modify rights to the selected folder, or the service installation will fail.

- **Service account** – Select the account type that the service is to use.
    - **Local System** – Select this to run the service as a Local System account that has extensive privileges on the local computer and in the network. This account type should not normally be used in a production environment, but it could be useful in a test environment. This account type acts as a computer on the network, and it does not have a password.
    - **Local Service** – Select this to run the service as a Local Service account that is specifically intended for services for security. This account type has minimum privileges on the local computer, so your computer is still relatively safe even if the service is compromised. The account cannot access most of the folders on the local computer by default, so you must explicitly give this account full control to each project folder that you want the service to be able to use. The account presents anonymous credentials on the network.
    - **Network Service** – Select this to run the service as a Network Service account. This is similar to a Local Service account in that it has minimum privileges on the local computer, so your computer is still relatively safe even if the service is compromised, and you must explicitly give it full control to the project folders. The differences are that this account type has the permissions for 'Log on as a service' and 'Impersonate a client after authentication' by default, and it identifies itself in the network by using the credentials of the computer account.
    - **Other Account** – Select this to run the service as a specific user account that requires a username and a password. Enter the required credentials in the **Account** (DOMAIN\user) and **Password** fields.

Tip: In production environments, the account type is usually set to either Local Service or Network Service. For more information on service accounts, see https://docs.microsoft.com/en-us/windows/win32/services/service-user-accounts.

- **Start on system startup** – Select this option if you want the service to be running whenever the computer is on. You can always start and stop the service as needed, using either the Web API management tools or Windows services.

4. Click **Install**. You are prompted if the installing succeeded.
5. You can now configure the service as described in Configuring a service and then start it as described in Starting a service.

2.2. Upgrading a service

If CADMATIC Plant/Outfitting has been upgraded to a new version, you can also upgrade the related Web API services to that version.

Do the following:

1. Open the **CADMATIC Web API Installer** dialog.
2. Select the service to be upgraded, and click **Upgrade**.
3. In the **Upgrade Service** dialog, review the following settings.
    - **Configuration directory** – Make sure the correct configuration folder is selected.
    - **Workstation profile** – Make sure the correct workstation profile is selected.
4. Click **OK**. You are prompted if the upgrade succeeded.

2.3. Uninstalling a service

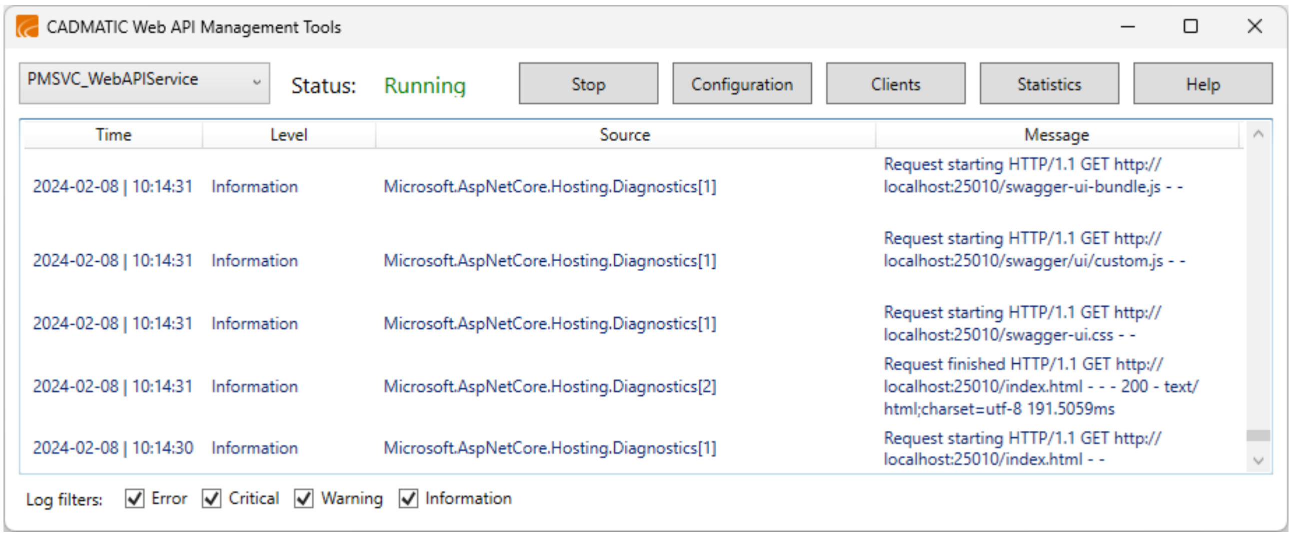
You can uninstall a CADMATIC Web API service that is no longer needed. This also deletes the Plant Modeller areas that the service has been using, and if you choose, the configuration files that the service has been using.

Do the following:

1. Open the **CADMATIC Web API Installer** dialog.
2. Select the service to be removed, and click **Uninstall**. You are prompted if the uninstalling succeeded.
3. You are prompted whether to delete the configuration files that the service has been using.

3. CADMATIC Web API Management Tools

You can use the **CADMATIC Web API Management Tools** dialog to manage installed Web API services.



Time	Level	Source	Message
2024-02-08 | 10:14:31	Information	Microsoft.AspNetCore.Hosting.Diagnostics[1]	Request starting HTTP/1.1 GET http://localhost:25010/swagger-ui-bundle.js - -
2024-02-08 | 10:14:31	Information	Microsoft.AspNetCore.Hosting.Diagnostics[1]	Request starting HTTP/1.1 GET http://localhost:25010/swagger/ui/custom.js - -
2024-02-08 | 10:14:31	Information	Microsoft.AspNetCore.Hosting.Diagnostics[1]	Request starting HTTP/1.1 GET http://localhost:25010/swagger-ui.css - -
2024-02-08 | 10:14:31	Information	Microsoft.AspNetCore.Hosting.Diagnostics[2]	Request finished HTTP/1.1 GET http://localhost:25010/index.html - - - 200 - text/html;charset=utf-8 191.5059ms
2024-02-08 | 10:14:30	Information	Microsoft.AspNetCore.Hosting.Diagnostics[1]	Request starting HTTP/1.1 GET http://localhost:25010/index.html - -

Log filters: ✓ Error ✓ Critical ✓ Warning ✓ Information

To access this dialog, do the following:

- Open the **CADMATIC Web API Installer** dialog, and click **Management Tools**.

3.1. Starting a service

You can start a properly configured Web API service using either the Web API management tools or Windows services. Starting a Web API service starts to generate a log that describes the actions that the service is performing. If there are critical errors, the service cannot start.

Prerequisites

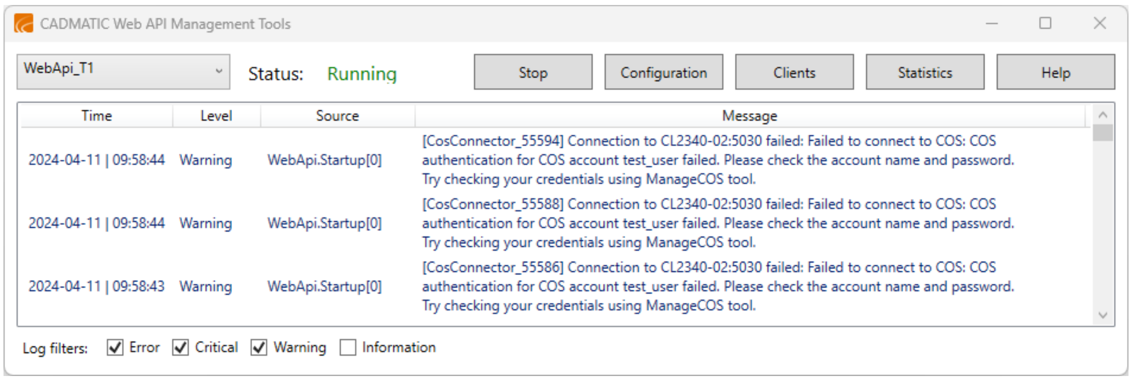
- The SSL certificate has been installed.
- The COS server and the user settings have been defined, as described in Configuring a service.

Do the following:

1. Open the **CADMATIC Web API Management Tools** dialog.
2. Select the required Web API service from the drop-down list, and click **Start**.

- If the service started, the status is set to "Running".
- If the service could not start, or it started but the log indicates that there are errors, analyze the log to see what might be the problem.
    - If the log indicates that HTTPS endpoints have not been set up, verify that the SSL certificate is correct.
    - If the log indicates that COS connection failed, verify the COS server and user settings in the service configuration.
    - If the log indicates a Plant Modeller connector error, click **Open Log** to open the Plant Modeller area log.

  You can filter the log pane by hiding unnecessary messages. For example, you may want to clear all other logging levels except critical errors.



Status: Running

Time	Level	Source	Message
2024-04-11 | 09:58:44	Warning	WebApi.Startup[0]	[CosConnector_55594] Connection to CL2340-02:5030 failed: Failed to connect to COS: COS authentication for COS account test_user failed. Please check the account name and password. Try checking your credentials using ManageCOS tool.
2024-04-11 | 09:58:44	Warning	WebApi.Startup[0]	[CosConnector_55588] Connection to CL2340-02:5030 failed: Failed to connect to COS: COS authentication for COS account test_user failed. Please check the account name and password. Try checking your credentials using ManageCOS tool.
2024-04-11 | 09:58:43	Warning	WebApi.Startup[0]	[CosConnector_55586] Connection to CL2340-02:5030 failed: Failed to connect to COS: COS authentication for COS account test_user failed. Please check the account name and password. Try checking your credentials using ManageCOS tool.

Log filters: [✓ Error] [✓ Critical] [✓ Warning] ☐ Information

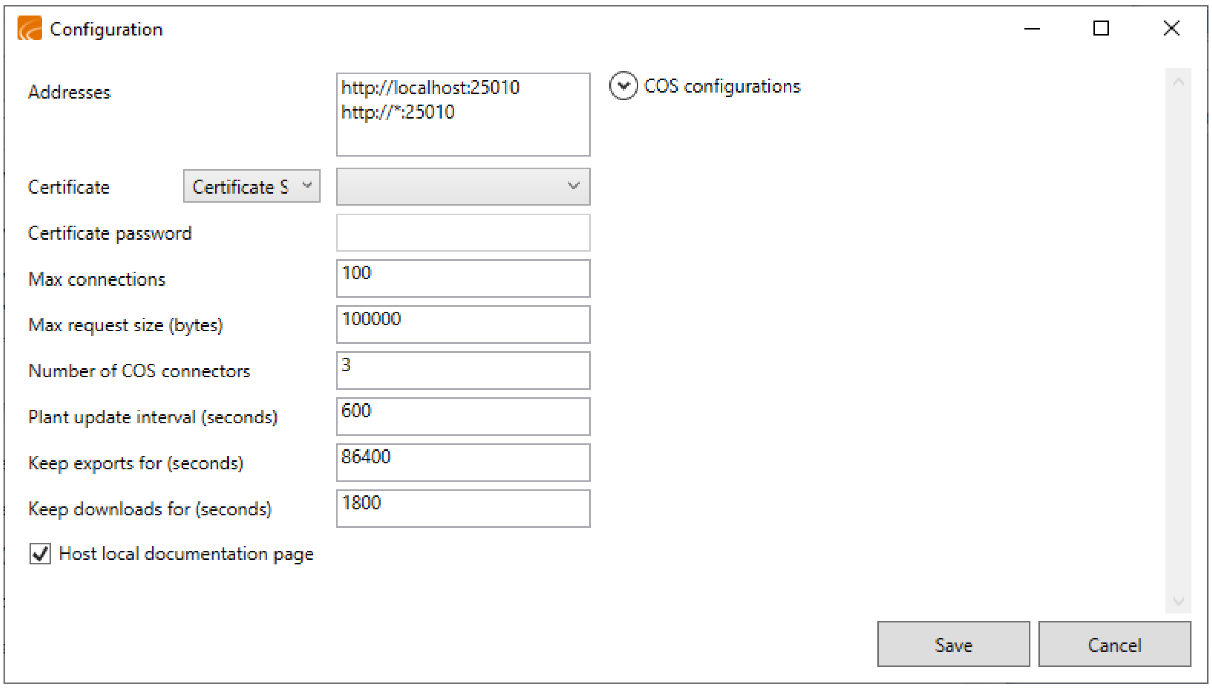
When a Web API service is running, you can select the service from the drop-down list and click **Stop** to stop the service. You must do this, for example, if you want to clear the list of calls to deprecated and sunset endpoints, as described in Viewing statistics.

3.2. Configuring a service

You can configure a Web API service as follows.

Do the following:

1. Open the **CADMATIC Web API Management Tools** dialog.
2. Select the required Web API service from the drop-down list, and click **Configuration**. The **Configuration** dialog opens.



Configuration

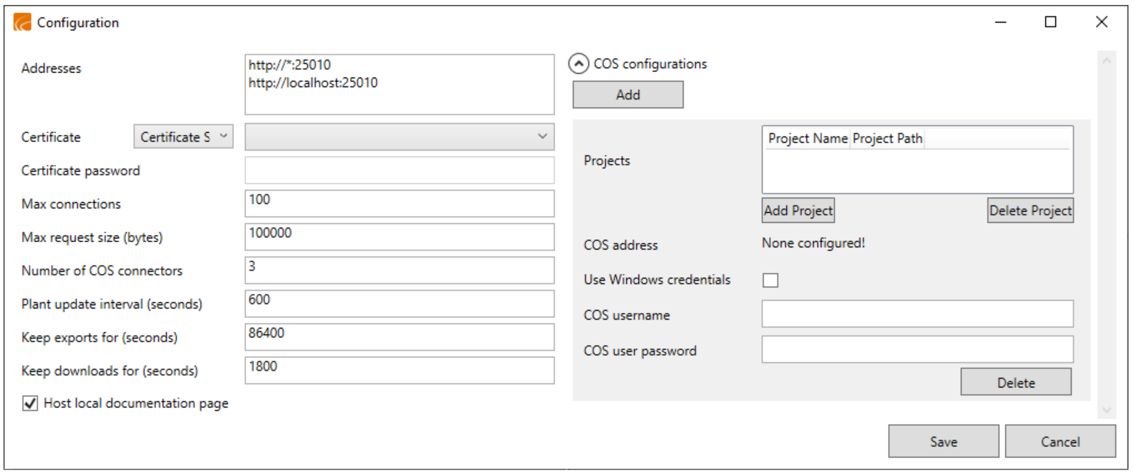
Addresses	http://localhost:25010 http://*:25010
Certificate	Certificate S
Certificate password	
Max connections	100
Max request size (bytes)	100000
Number of COS connectors	3
Plant update interval (seconds)	600
Keep exports for (seconds)	86400
Keep downloads for (seconds)	1800

☑ Host local documentation page

3. Define the general settings:
   - **Addresses** – Enter the URLs from which connections are accepted, in the format **<protocol>://<address>:<port>**. Each new URL must be entered on a separate row.
     - Protocol: HTTP or HTTPS. Use HTTPS if you want to ensure that data is securely transmitted (requires a certificate).
     - Address: IP address or network name. If you set this to *localhost*, only connections from the local machine will be accepted. The asterisk character allows connections from any address: *http://*:25010*
     - Port: Any unused port.
   - **Certificate** – Select the certificate from the certificate store or from the file system. The type of the certificate is project-specific. Follow your organization's guidelines for obtaining the certificate. If you do not have a certificate, https:// connections will not work.
   - **Certificate password** – Enter the password of the selected certificate.
   - **Max connections** – Define the maximum number of connections the API accepts at a time. The default value is 100.
   - **Max request size (bytes)** – Define the maximum size of a single request. The default size is 100 MB.

- **Number of COS connectors** – Define the maximum number of concurrent COS connectors to use for handling the requests. If there is more than one connector and the originally assigned connector is occupied, another connector can take over the request. The default number is 3.
- **Plant update interval (seconds)** – Define the interval for updating the Plant/Outfitting data. The default is 600 seconds.
- **Keep exports for (seconds)** – Specify how long to keep exported files until they are deleted. The default is 86400 seconds
- **Keep downloads for (seconds)** – Specify how long to keep downloaded files until they are deleted. The default is 1800 seconds.
- **Host local documentation page** – Select whether to allow the Swagger UI page to be used for testing the Web API in local installations. See Swagger UI.

4. Define how the service is to connect to projects in COS:
   a. Expand **COS configurations** and click **Add**. The COS connection settings are displayed.


Configuration

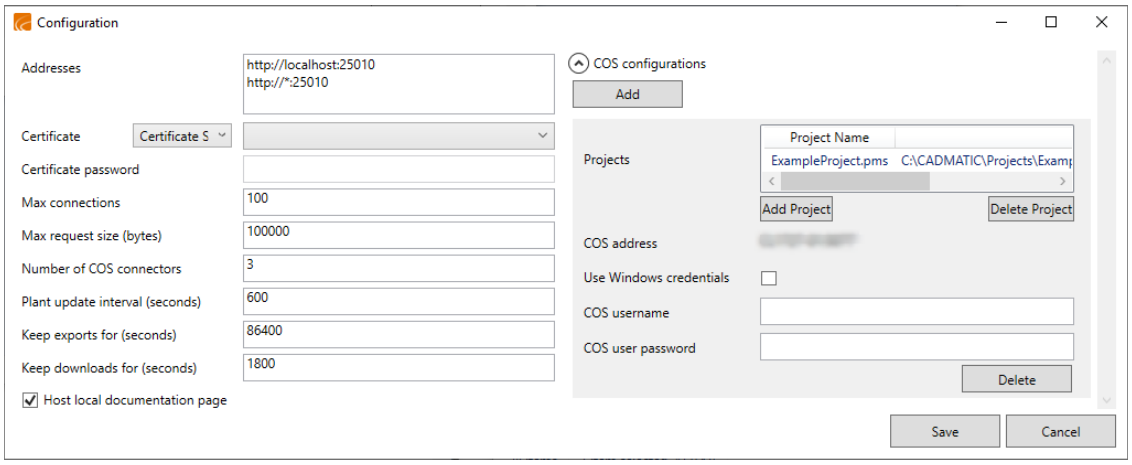
Addresses	http://*:25010 http://localhost:25010
Certificate	Certificate S
Certificate password	
Max connections	100
Max request size (bytes)	100000
Number of COS connectors	3
Plant update interval (seconds)	600
Keep exports for (seconds)	86400
Keep downloads for (seconds)	1800
☑ Host local documentation page	

COS configurations

Projects	Project Name, Project Path
COS address	None configured!
Use Windows credentials	☐
COS username	
COS user password	

   b. To add a project, click **Add Project** and browse to the **<projectdbname>.pms** folder that contains the project database . The **COS address** field shows the address of the COS server, and you can add more projects that use the same COS server.
   c. To define how to authenticate the service in COS, do the following:
      - If the service was installed as Local System, Local Service or Network Service, enter the COS username and password to use.
      - If the service was installed as Other Account, you can select the **Use Windows credentials** option to use the same Windows account that the service is using also for accessing COS. This requires that the Windows user account is a valid

COS user account. Otherwise, clear the option and specify the COS username and the COS password to use.



Addresses	http://localhost:25010 http://*:25010
Certificate	Certificate S
Certificate password	
Max connections	100
Max request size (bytes)	100000
Number of COS connectors	3
Plant update interval (seconds)	600
Keep exports for (seconds)	86400
Keep downloads for (seconds)	1800
☑ Host local documentation page	

Project Name	
ExampleProject.pms	C:\CADMATIC\Projects\Examp

COS address	[redacted]
Use Windows credentials	☐
COS username	
COS user password	

5. Click **Save**.
6. You can now start the service, as described in Starting a service.

3.3. Defining clients for a service

Only predefined clients with valid access tokens can access the CADMATIC Web API endpoints.

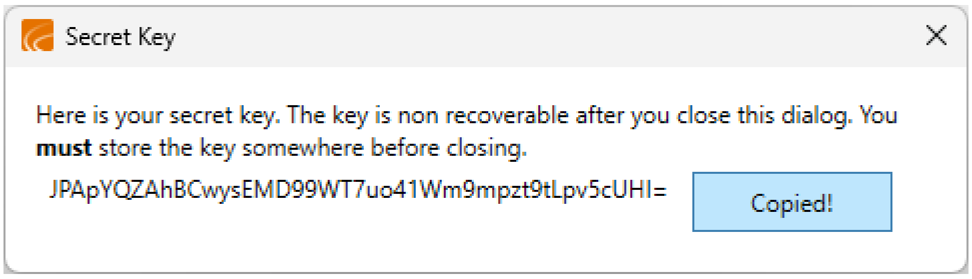
3.3.1. Adding clients

Create a client for every human user or software program that must be able to access the Web API.

Important: This procedure provides you with the Name, the Client ID, and the Secret Key of the new client. You will need this information when creating an access token for the client.

Do the following:

1. Open the **CADMATIC Web API Management Tools** dialog.
2. Select the required Web API service from the drop-down list, and click **Clients**.
3. In the **Clients** dialog, click **Add Client**.
4. In the **Client Name** dialog, enter a name (5–256 characters, spaces are allowed) for the client, and click **OK**.
5. In the **Secret Key** dialog, click **Copy** to copy the Secret Key to the clipboard.



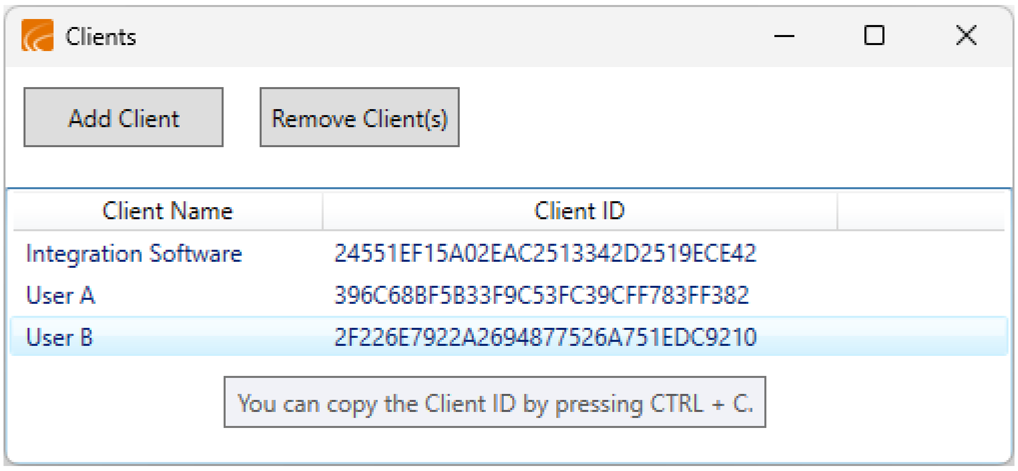
Secret Key

Here is your secret key. The key is non recoverable after you close this dialog. You **must** store the key somewhere before closing.

JPApYQZAhBCwysEMD99WT7uo41Wm9mpzt9tLpv5cUHI=

Important: Store the key in a safe place. Once you close this dialog, there is no way to recover the key.

6. In the **Clients** dialog, you can see the Name and the Client ID of the new client. You can now create an access token for the new client, for example, as described in Creating access tokens.



Clients

Client Name	Client ID
Integration Software	24551EF15A02EAC2513342D2519ECE42
User A	396C68BF5B33F9C53FC39CFF783FF382
User B	2F226E7922A2694877526A751EDC9210

You can copy the Client ID by pressing CTRL + C.

3.3.2. Removing clients

You can remove clients that should no longer be able to access the Web API.

Do the following:

1. Open the **CADMATIC Web API Management Tools** dialog.
2. Select the required Web API service from the drop-down list, and click **Clients**.
3. In the **Clients** dialog, select the clients to be removed from the list, and click **Remove Client(s)**.

3.4. Viewing statistics

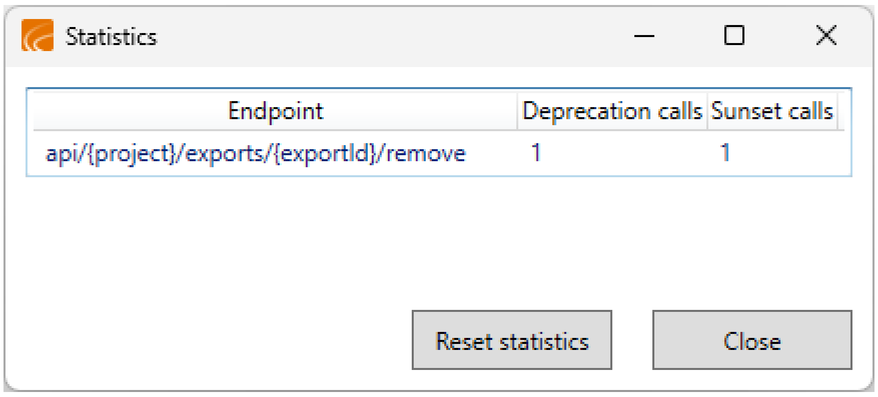
You can verify if any calls have been made to endpoints that are either deprecated or sunset. If such calls exist, determine whether newer endpoints that perform similar functions are available, and update your code to use these newer endpoints. The formal process of first deprecating, then sunsetting, and finally removing endpoints is described in Endpoint deprecation process.

Prerequisites

- If you intend to reset the statistics, you must stop the Web API service.

Do the following:

1. Open the **CADMATIC Web API Management Tools** dialog.
2. Select the required Web API service from the drop-down list, and click **Statistics**. The **Statistics** dialog opens.



Endpoint	Deprecation calls	Sunset calls
api/{project}/exports/{exportId}/remove	1	1

3. If the dialog lists any calls to deprecated or sunset endpoints, make a note of those calls so that you can start planning when to stop using them.
4. You can clear the list by clicking **Reset statistics**. If the service is running, you are prompted that it must be stopped first.
5. Click **Close**.

4. CADMATIC Web API Endpoints

This section provides information on using the endpoints of a running CADMATIC Web API service.

4.1. Endpoint documentation

You can browse endpoint documentation on the Web API reference page or via Swagger UI.

4.1.1. Web API reference

The Web API Reference is a static reference page that has been generated directly from the API itself. It describes all the endpoints and their parameters, enabling you to access the API documentation even when the Web API service is not readily available.

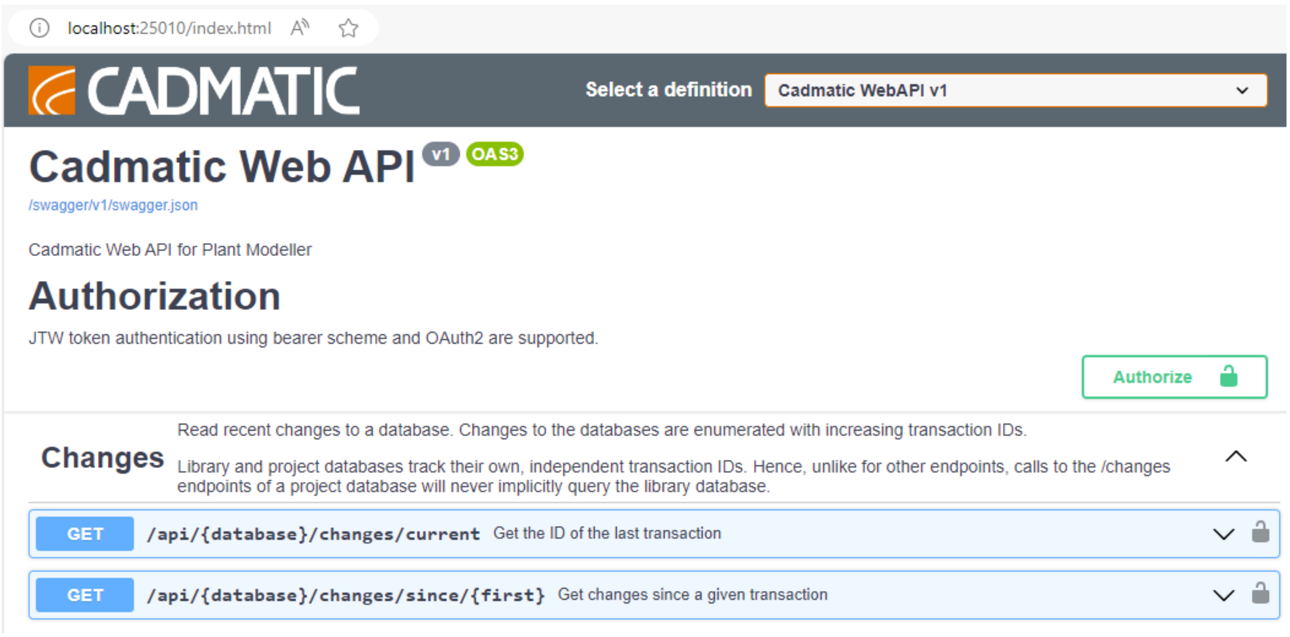
You can open the reference page in a web browser from this link:
https://docs.cadmatic.com/webapi/2024T3/Content/Topics/WebAPI_reference.html

4.1.2. Swagger UI

Swagger is an open-source toolkit that helps developers design, build, and document REST APIs. A Swagger page, or Swagger UI, provides a user-friendly interface to the API, enabling developers to understand its capabilities, interact with its endpoints, and test it directly from the web browser without needing to write code. Thus, it serves as both an interactive API documentation and a testing tool.

You can open the Swagger UI page in a web browser by entering the URL that was used to configure the Web API service.

On the Swagger page, you can browse the available endpoints. For each endpoint, it shows the applicable HTTP method, a description, a list of parameters, responses and examples. The required parameters are indicated in the Web API reference page.


localhost:25010/index.html

CADMATIC

Select a definition: Cadmatic WebAPI v1

Cadmatic Web API v1 OAS3

/swagger/v1/swagger.json

Cadmatic Web API for Plant Modeller

Authorization

JTW token authentication using bearer scheme and OAuth2 are supported.

Authorize

Changes

Read recent changes to a database. Changes to the databases are enumerated with increasing transaction IDs.

Library and project databases track their own, independent transaction IDs. Hence, unlike for other endpoints, calls to the /changes endpoints of a project database will never implicitly query the library database.

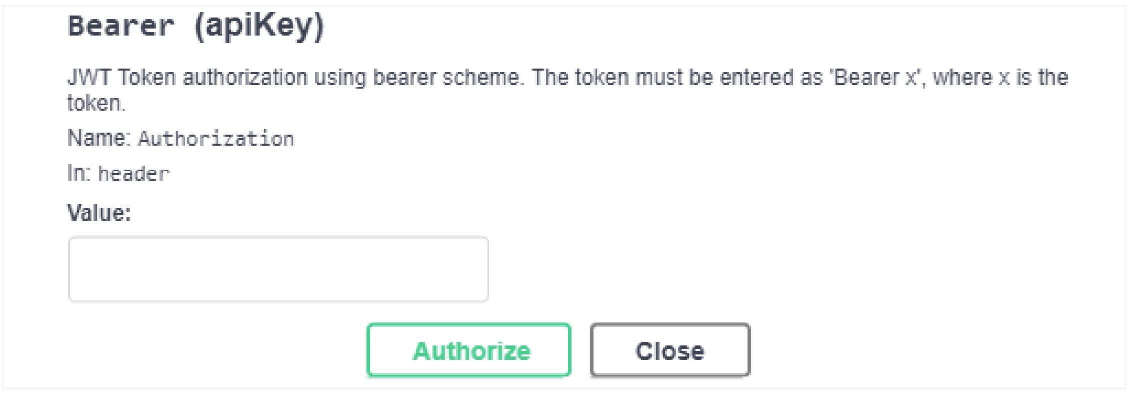
GET `/api/{database}/changes/current` Get the ID of the last transaction

GET `/api/{database}/changes/since/{first}` Get changes since a given transaction

If you have the information that is required for authenticating your client, you can also test the endpoints on the Swagger page.

Do the following:

1. Enter the URL of the Web API in a web browser. The Swagger page opens.
2. Click Authorize. The **Available authorizations** view opens.
3. Enter the authentication information, and click the Authorize button of the given authorization method.
   - To use JSON authorization, enter the value as *Bearer x* where x is the token.


Bearer (apiKey)

JWT Token authorization using bearer scheme. The token must be entered as 'Bearer x', where x is the token.

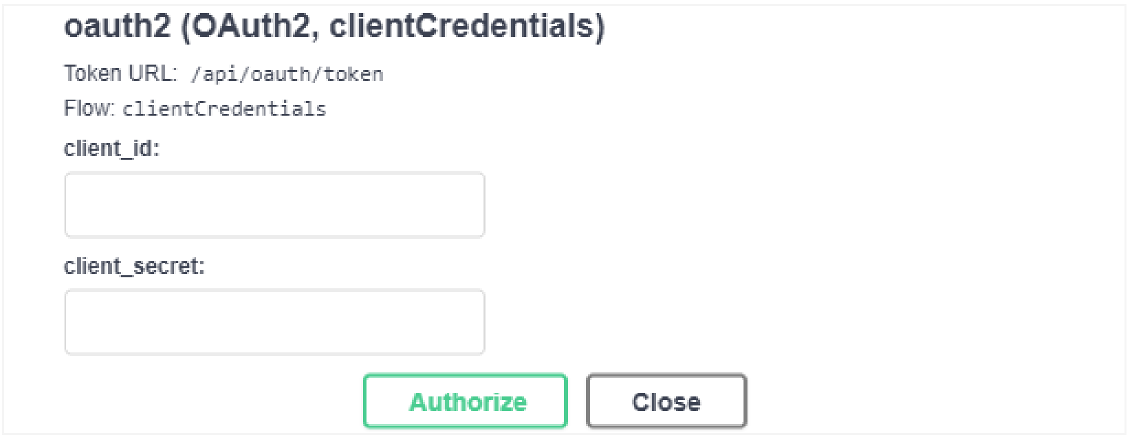
Name: Authorization

In: header

Value:

Authorize Close

- To use OAuth 2.0 Client Credentials, enter the Client ID and the Secret Key.



oauth2 (OAuth2, clientCredentials)

Token URL: `/api/oauth/token`

Flow: `clientCredentials`

client_id:

client_secret:

Once authorization is granted, you can try out the endpoint functions on the Swagger page.

4.2. Targeting databases in query strings

The CADMATIC Web API can be used for various queries whose target is library or project data. Some endpoints can target either library or project databases, while others only target project databases. However, it is important to note that even when the target is the project database, responses may still include objects derived from the library database.

4.2.1. Target database identification

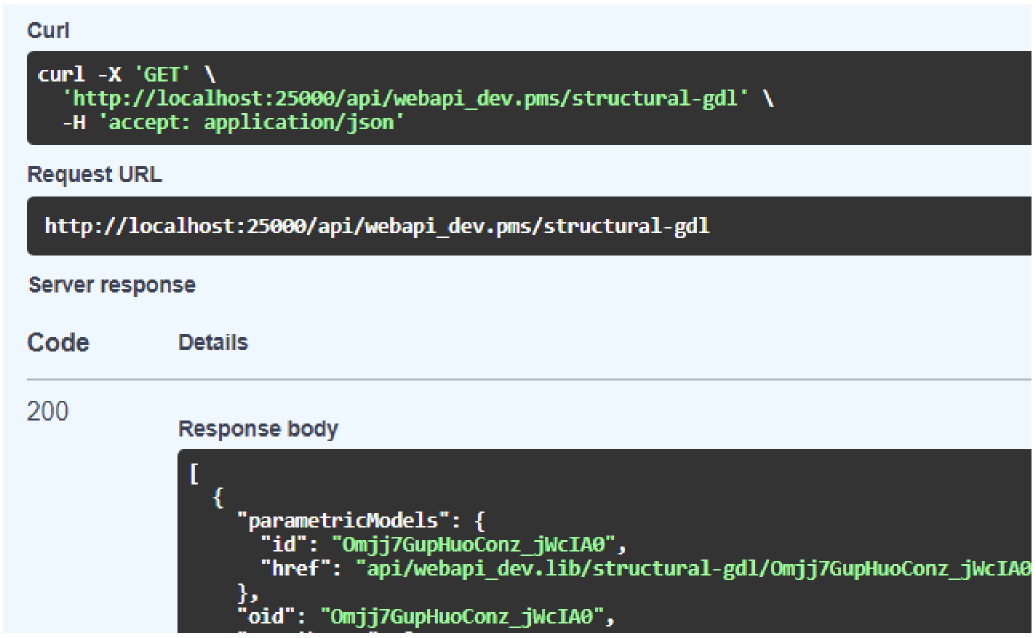
In the CADMATIC Web API, queries that target a project or library database must have a database identifier string as part of their URL.

- *{project}* in the endpoint path means that the target must be a project database:
  `GET .../{project}/exports`
- *{database}* in the endpoint path means that the target can be either a library or project database:
  `GET .../{database}/equipment-gdl`

If the database name is unique in the given environment, the identifier string only needs to specify the name.

- Specify the name of the target database in the format *dbName.dbExtension*.

  In this example, there is only one database with the name *webapi_dev.pms*, so that works as the identifier:



Curl

```
curl -X 'GET' \
  'http://localhost:25000/api/webapi_dev.pms/structural-gdl' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:25000/api/webapi_dev.pms/structural-gdl
```

Server response

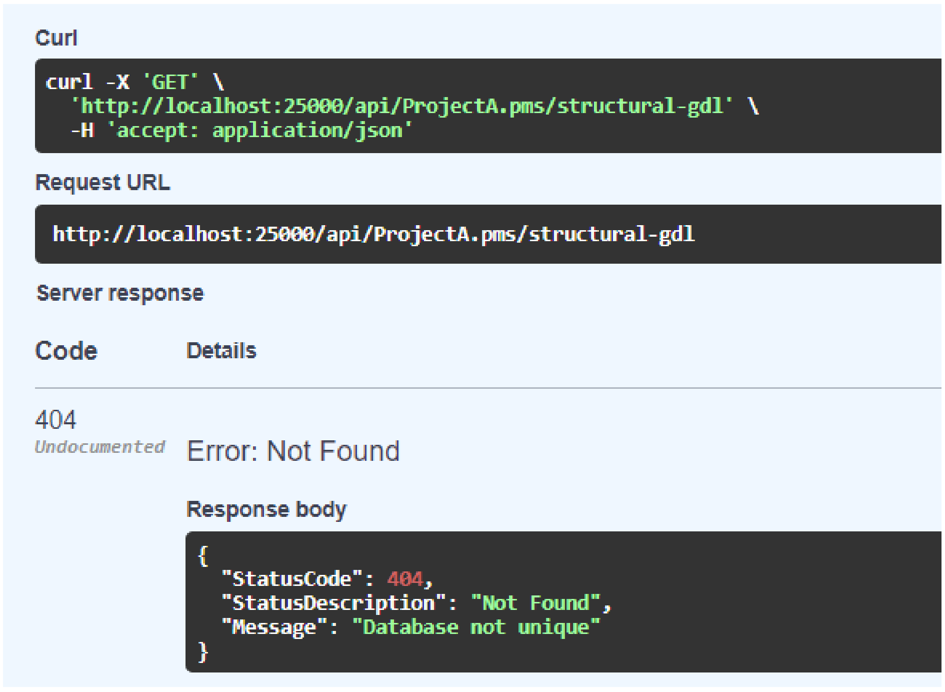
Code Details

200

Response body

```
[
  {
    "parametricModels": {
      "id": "Omjj7GupHuoConz_jWcIA0",
      "href": "api/webapi_dev.lib/structural-gdl/Omjj7GupHuoConz_jWcIA0
    },
    "oid": "Omjj7GupHuoConz_jWcIA0",
```

In this example, the response indicates that *ProjectA.pms* is not a unique database name:



Curl

```
curl -X 'GET' \
  'http://localhost:25000/api/ProjectA.pms/structural-gdl' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:25000/api/ProjectA.pms/structural-gdl
```

Server response

Code Details

404
Undocumented

Error: Not Found

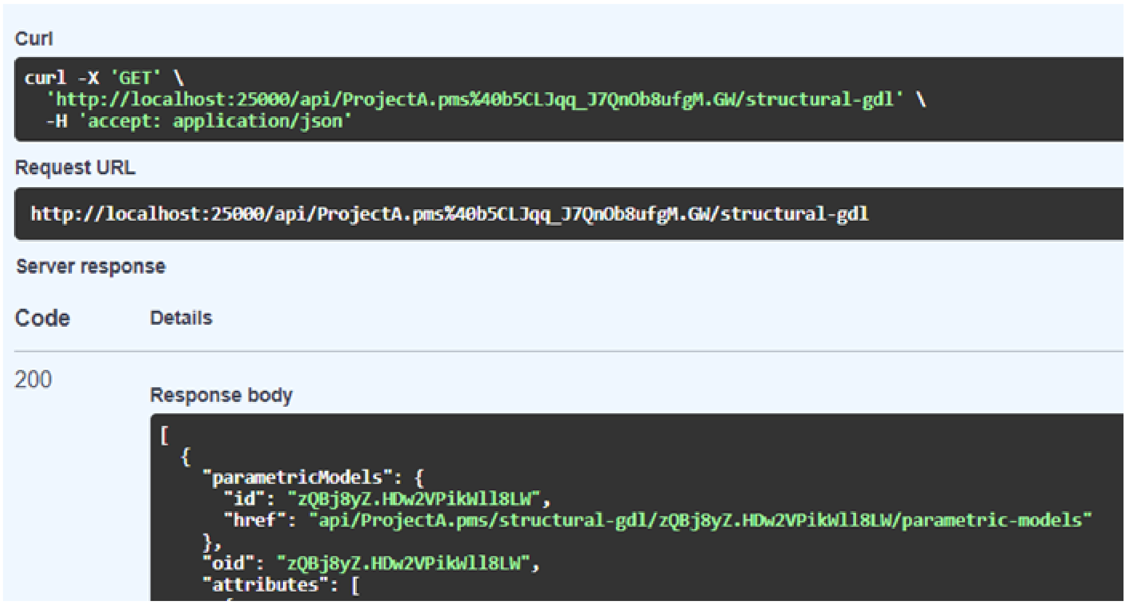
Response body

```
{
  "StatusCode": 404,
  "StatusDescription": "Not Found",
  "Message": "Database not unique"
}
```

If the database name is not unique in the given environment, you must also identify the server. This can happen especially in replicated projects where both the master server and replica servers are connected to the Web API, and replicated databases have inherited their name from the master database.

- Specify the name and the location of target database in the format *dbName.dbExtension@serverID* where *serverID* is one of the following: server name (might not be unique), server address (guaranteed to be unique), or server COS OID (practically guaranteed to be unique).

  In this example, the database is identified by its name and the COS OID of the server:



Curl

```
curl -X 'GET' \
  'http://localhost:25000/api/ProjectA.pms%40b5CLJqq_J7QnOb8ufgM.GW/structural-gdl' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:25000/api/ProjectA.pms%40b5CLJqq_J7QnOb8ufgM.GW/structural-gdl
```

Server response

Code Details

200

Response body

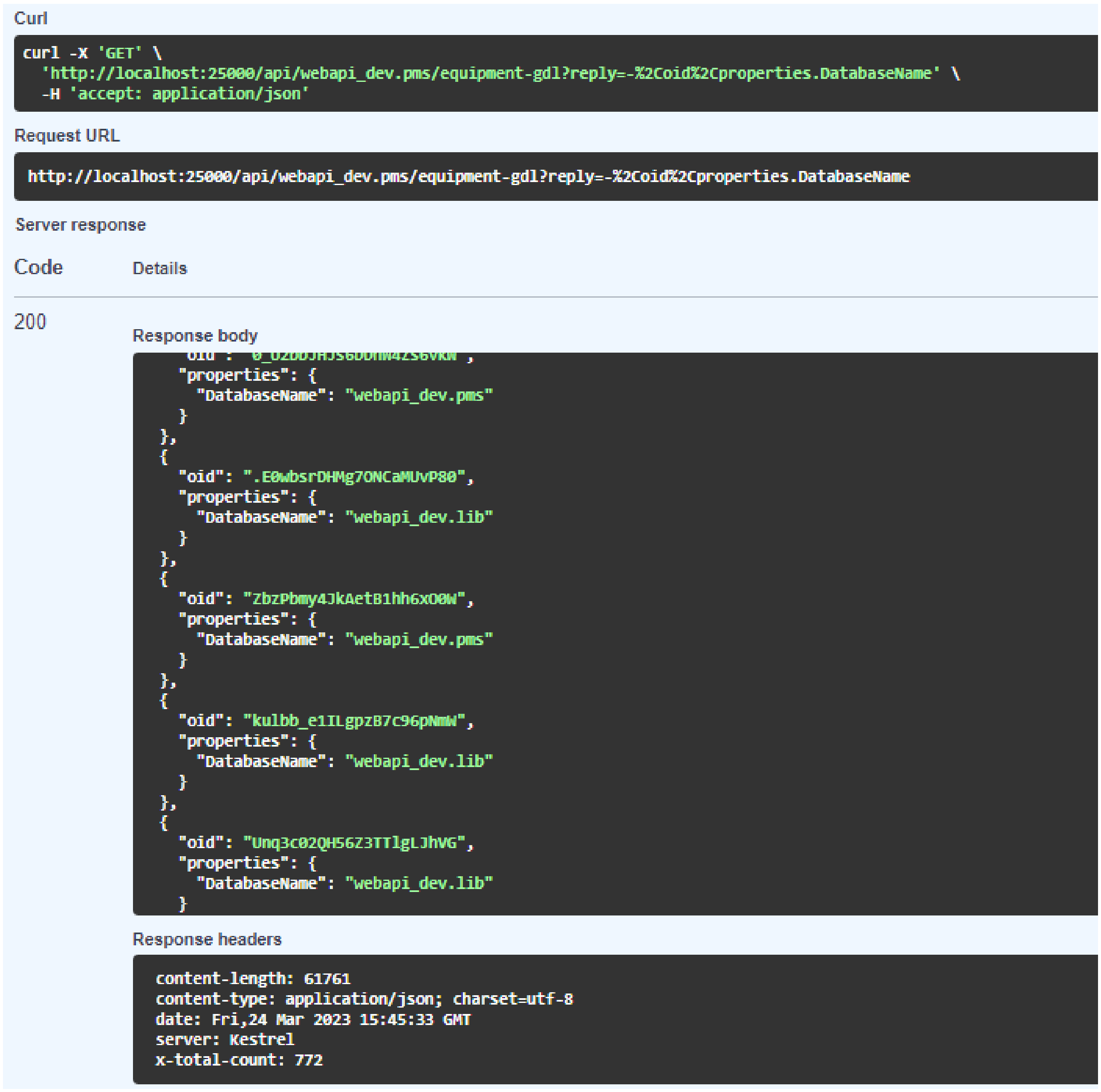
```
[
  {
    "parametricModels": {
      "id": "zQBj8yZ.HDw2VPikWl18LW",
      "href": "api/ProjectA.pms/structural-gdl/zQBj8yZ.HDw2VPikWl18LW/parametric-models"
    },
    "oid": "zQBj8yZ.HDw2VPikWl18LW",
    "attributes": [
```

4.2.2. Implicit library database queries

In many cases, you are not interested in whether the project data is stored in the library or the project database—you just want to know what data is available for the design project. Therefore, when you are using an endpoint whose target can be a library or project database, and your query is targeting the project database, the operation automatically performs an implicit library query. This means that the query returns results from the project database, but also from the library database.

Endpoints that require the target to be a project database do not implicitly query the library database.

The following example shows a request that primarily reads all objects from a project database (*webapi_dev.pms*), but the response also includes objects from the library database (*webapi_ dev.lib*) that the project is using. (Image taken from the Swagger UI page, with access token missing.)



Curl

```
curl -X 'GET' \
  'http://localhost:25000/api/webapi_dev.pms/equipment-gdl?reply=-%2Coid%2Cproperties.DatabaseName' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:25000/api/webapi_dev.pms/equipment-gdl?reply=-%2Coid%2Cproperties.DatabaseName
```

Server response

Code Details

200

Response body

```
      "oid": "0_U2DDJHJS6DDhW4ZS6VkW",
      "properties": {
        "DatabaseName": "webapi_dev.pms"
      }
    },
    {
      "oid": ".E0wbsrDHMg7ONCaMUvP80",
      "properties": {
        "DatabaseName": "webapi_dev.lib"
      }
    },
    {
      "oid": "ZbzPbmy4JkAetB1hh6xO8W",
      "properties": {
        "DatabaseName": "webapi_dev.pms"
      }
    },
    {
      "oid": "kulbb_e1ILgpzB7c96pNmW",
      "properties": {
        "DatabaseName": "webapi_dev.lib"
      }
    },
    {
      "oid": "Unq3c02QH56Z3TTlgLJhVG",
      "properties": {
        "DatabaseName": "webapi_dev.lib"
      }
```

Response headers

```
content-length: 61761
content-type: application/json; charset=utf-8
date: Fri,24 Mar 2023 15:45:33 GMT
server: Kestrel
x-total-count: 772
```

The same mechanism is also applied to operations that target individual objects: you can read, modify, and delete objects with implicit library queries. If the targeted object is located in */{library}/objecttype/oid*, you can access it also via */{project-using-the-library}/objecttype/oid*.

However, creating a new object always creates it in the specified database. If you create a new object under */{library}/objecttype*, then it is created in the library database. If you create a new object under */{project-using-the-library}/objecttype*, then it is created in the project database.

4.3. Using filtering in query strings

Calls requesting data from endpoints may include a query string that filters the data to make read operations quicker and to decrease the size of the response.

The CADMATIC Web API supports the following filtering methods: query filters, reply filters, and pagination. Query filters and reply filters are supported on all GET endpoints that return a collection of items, while pagination is only supported on selected GET endpoints.

4.3.1. Query filters

You can add query filters to the query string to enable the API call to target only items that have a given property, and optionally, a given value in that property.

You can include multiple query filters or use both query filters and reply filters in the same call. If the call includes multiple query filters, the results from the individual queries are combined using a logical AND operator. Combining multiple query filters with OR is not directly supported, but you can perform multiple GET requests and combine the results on the client side.

4.3.1.1. Syntax of query filters

Query filters can be constructed using the following syntax:

```
<neg><jsonfield>|<jsondictfield>|attributes(<string>)<operator><value>
```

where

- `<neg>` is an optional exclamation mark (!) that causes each condition to be negated and evaluated separately. Negation is most useful when the filter does not include an optional operator–value condition.
- `<jsonfield>|<jsondictfield>|attributes(<string>)` specifies the targeted field in the response. The query checks whether the field exists and evaluates for false if it does not.
  - `<jsonfield>` specifies the targeted JSON property using dot notation: `object.properties.objectType="Drawing"`
  - `<jsondictfield>` specifies the targeted JSON property using bracket notation: `[<string>]`. You can also mix dictionary like access and the `<jsonfield>` path: `["object"].properties["objectType"]="Drawing"`
  - `attributes(<string>)` specifies a COS attribute tag as `"<string>"`. If the string

itself contains double-quote (") or backslash (\) characters, they must be escaped with the backslash (\) character.

- `<operator><value>` defines an optional condition of the targeted field. If the operator is specified, then also the value must be given.
    - `<operator>` can be one of the following:

=	equal to
!=	not equal to
<=	less than or equal to
<	less than
>=	greater than or equal to
>	greater than
R	regular expression
in	included in the given set
notin	not included in the given set

Note: The use of comparison operator requires the value to be a string, a number, or a Boolean. Other types of values cause the filter to be ignored.

-
    - `<value>` can be one of the following, depending on the data type of the JSON field:

string	A quoted string.
number	A quoted string, or just the number without the quotes.
Boolean	`"true"` or `"false"`
list	A comma-separated list of items inside square brackets, where each item is a quoted string or a number: `[<item1>, <item2>, …, <itemn>]`

4.3.1.2. Rules for applying query filters

If the filter does not have an operator–value condition, it only checks if the object has the given field.

- *?query=properties.ModificationTime*
  The filter finds objects that have the 'ModificationTime' property.

If the filter has an operator–value condition, it checks if the field exists and if its value satisfies the given condition.

- *?query=properties.ModificationTime>1645034215*
  The filter finds objects where 'ModificationTime' is greater than the given value.

If the filter is negated with a leading exclamation mark, it evaluates each part of the filter expression separately.

- *?query=!properties.ModificationTime*
  The filter finds objects that do not have the 'ModificationTime' property.
- *?query=!properties.ModificationTime>1645034215*
  The filter finds (a) objects that do not have the 'ModificationTime' property and (b) objects where 'ModificationTime' is not greater than the given value.

If the filter contains an element that is an array, the rest of the filter's path is used to evaluate every item in the array. If at least one of the items matches the filter, the object is considered a match.

- *?query=connectionPoints.data.tg1*
  The filter finds objects where any of the connection points has the key 'tg1' in its data.
- *?query=!connectionPoints.data.tg1*
  The filter finds objects where none of the connection points has the key 'tg1' in its data.
- *?query=!connectionPoints["data"]["a.b"]*
  The filter uses bracket notation to find objects where none of the connection points has the key 'a.b' in its data. The filter cannot use dot notation because the key contains a dot, which is a special character.

If the filter tries to compare the value of an element that is an array, the filter is ignored.

- *?query=connectionPoints=5*
  The filter is ignored. An array cannot equal a number or a string.

4.3.2. Reply filters

You can add reply filters to the query string to enable the API call to remove data fields that are not needed in the response, as well as to put fields back into it.

You can include multiple reply filters or use both reply filters and query filters in the same call.

4.3.2.1. Syntax of reply filters

Reply filters can be constructed using the following syntax:

```
<replyfilter>, <replyfilter>, … , <replyfilter>
```

where each `<replyfilter>` can define the elements `<exclusionSpecifier><jsonfield>|attributes(<string>)` as follows.

- `<exclusionSpecifier>` can be either '-' or '+', and it can target all the fields or just a specific field.
  - `-<fieldName>` removes the specified field from the response.
  - `–` removes all the fields from the response. If none are put back, the object remains as an empty object.
  - `+<fieldName>` (or just `<fieldName>`) adds the specified field back into the response.
  - `+` adds all the fields back into the response.
- `<jsonfield>` specifies the targeted JSON field using dot notation.
  Because attributes are not hard-coded special fields, but rather runtime values, they cannot be specified by the <jsonfield> syntax.
- `attributes(<string>)` specifies attributes in the targeted field. `<string>` is a double-quoted string that contains the tag name of some COS attribute. If the string itself contains double-quote (") characters, they must be escaped with the backslash (\) character.

4.3.2.2. Rules for applying reply filters

If the filter contains an exclusion specifier, the filter can remove fields from the response or put fields back into it.

- *?reply=-properties*
  The filter removes all property fields from the response.
- *?reply=+properties.foobar*
  The filter puts the 'foobar' property back into the response.

If the call includes more than one reply filter, the filters are applied in the order that they are given.

- *?reply=-,foobar*

  If an object in the response has these properties:
  ```
  {
    "foo" : "bar"
    "bar" : 123
    "foobar": [1, 2, 3]
  }
  ```
  the first filter removes every property from the object and the second filter puts the 'foobar' property back into it:
  ```
  {
    "foobar": [1, 2, 3]
  }
  ```
  (Alternatively, the same result could have been achieved by *?reply=-foo,-bar*.)

If the path of the reply filter includes a field that is an array, the filter is applied to all the items in the array.

- *?reply=-foobar.a*

  If an object in the response has this property:
  ```
  {
    "foobar": [{ "a": 1 }, {"b": 2}, {"a": 1}]
  }
  ```
  the filter removes all the 'a' fields from the property:
  ```
  {
    "foobar": [{}, {"b": 2}, {}]
  }
  ```

4.3.3. Pagination

You can add pagination to the query string to start reading the response from a specific object (numeric offset, key-based offset), and to read only a subset of the objects from the response (limiting).

Numeric offset

In numeric offset, the 'offset' parameter specifies the zero-based index of the first object to be read (that is, how many objects to skip before starting to read).

GET api/Project.pms/catalogParts?offset=10

Key-based offset

In key-based offset, the 'after' parameter specifies the COS OID of the object after which to start the reading.

GET api/Project.pms/catalogParts?after=GkbG.pzZHiACC6j4RQYmE0

Tip: Key-based offset provides better performance than numeric offset.

Limiting

In limiting, the 'limit' parameter defines the maximum number of objects to be read from the set.

GET api/Project.pms/catalogParts?limit=50

Combining an offset and limiting

You can use an offset parameter and limiting in the same query string. In the following example, the first query string specifies a limit of 50 objects, which is used to get the OID of the last item in that subset ('GkbG.pzZHiACC6j4RQYmE0' in our example), and then another query string reads the next 50 objects. This process can be repeated until all items have been read; this is when the returned list is either empty or contains fewer items than the specified limit.

GET api/Project.pms/catalogParts/?limit=50

GET api/Project.pms/catalogParts/?after=GkbG.pzZHiACC6j4RQYmE0&limit=50

Combining pagination and query filters

You can use pagination together with Query filters, but note the following:

- The 'offset' and 'limit' parameters are applied to the filtered list, so their effect depends on the query filter.
- The 'after' parameter is applied before filtering, and it works even if the object that the parameter refers to is filtered out.

Page consistency

Reading the same page again keeps the same order of items, but the list of items might change.

- If an object has been removed from COS, re-reading the page removes the object from the page.
- If a new object has been added into COS, re-reading the page adds the object to the page, in accordance with the existing order of items.

Therefore, clients must not assume that re-reading a page will always return exactly the same results.

4.3.4. Filtering examples

The following examples demonstrate the use of query filters and reply filters.

4.3.4.1. Example 1: Read catalog parts with any author

You can use a query filter to find entities that have a certain property, regardless of the values of that property.

Task: Get all catalog parts that have the 'Author' attribute (tag 'U00' in our examples) from the project database ('Project.pms' in our examples).

Query string:

?query=attributes("U00")

Example query:

GET api/Project.pms/catalogParts/?query=attributes("U00")

4.3.4.2. Example 2: Read catalog parts with specific author

You can use a query filter to find entities that have a certain property set to a certain value.

Task: Get all catalog parts that have the 'Author' attribute set to 'Cadmatic User' from the project database.

Query string:

?query=attributes("U00")="Cadmatic User"

Example query:

GET api/Project.pms/catalogParts/?query=attributes("U00")="Cadmatic User"

4.3.4.3. Example 3: Read catalog parts with no author

You can use a query filter to find entities that do not have a certain property.

Task: Get all catalog parts that do not have the 'Author' attribute from the project database.

Query string:

?query=!attributes("U00")

Example query:

GET api/Project.pms/catalogParts/?query=!attributes("U00")

4.3.4.4. Example 4: Read changes made to model objects

You can use a logical AND to combine multiple query strings.

Task: Get all changes that have been made to model objects in the project database.

Query string:

?query=objectType="ModelObject"&query=transactionType="Modified"

Example query:

GET api/Project.pms/changes/?query=objectType="ModelObject"&query=transactionType="Modified"

4.3.4.5. Example 5: Exclude properties and references from catalog parts

You can use a reply filter to remove unnecessary fields from the JSON response.

Task: Get all catalog parts from the project database. Save bandwidth by excluding all of their properties and references to model objects from the response.

Query string:

?reply=-properties,-referencedObjects

Example query:

GET api/Project.pms/catalogParts/?reply=-properties,-referencedObjects

4.3.4.6. Example 6: Exclude properties, while keeping some fields

You can use a reply filter to get only some sub-properties from the properties structure of the JSON response. You can do this by first removing all the properties and then putting back only the necessary properties.

Task: Include the creation time of the object, but exclude all other properties from the response.

Query string:

?reply=-properties,properties.creationTime

4.3.4.7. Example 7: Combine query and reply filters

You can use query filters and reply filters in the same query string.

Task: Get all catalog parts created by a specific author within the last two days. Times must be given as a Unix timestamp; for the purposes of this example, we assume that timestamp '1673524566' represents some time from two days ago. Exclude all properties, including the 'Author' attribute, from the response (the author is the same for every part).

First, let's create the reply filter string:

?reply=-properties,-attributes("U00")

Then, let's create query filter strings for specific values of 'Author' and 'CreationTime':

?query=attributes(“U00”)="Cadmatic User"

?query=properties.CreationTime>=1673524566

Finally, let's combine the reply filter and the query filters into one query string:

?reply=-properties,-attributes("U00")&query=attributes("U00")="Cadmatic User"?query=properties.CreationTime>=1673524566

So, this query returns all catalog parts that were created in the last two days and where 'Author' is set to 'Cadmatic User'. The response itself will not include the 'Author' attribute or any of the properties since the Web API was explicitly told to remove them from the response.

4.3.4.8. Example 8: Use set operations

You can use the 'in' and 'notin' operators in query filters to match a property against a set of values, instead of making a separate query for each possible value.

Task: Get the objects whose COS object type is or is not 'Model Object' or 'Catalog Part'.

Query strings:

?query=properties.ObjectType in ["Model Object", "Catalog Part"]

?query=properties.ObjectType notin ["Model Object", "Catalog Part"]

4.3.4.9. Example 9: Use regular expressions

You can use regular expressions in query filters to find items that have specific property values.

Task: Get the Systems where the 'Name' attribute (tag '.dD') contains the substring 'water' or 'Water'.

Query string:

> *?query=Attributes(".dD")=R"water|Water"*

4.3.4.10. Example 10: Use dictionary like access

You can use bracket notation when the JSON field name includes characters that are not compatible with the standard dot notation. For example, COS attribute tags can contain dots.

Task: Get the entities whose creation time is at least a given point in time.

Query string, bracket notation:

> *?query=properties["CreationTime"] >= 1673524566*

Query string, dot notation:

> *?query=properties.CreationTime >= 1673524566*

4.4. Endpoint deprecation process

New features are continually being added to the CADMATIC Web API, which sometimes leads to existing endpoints becoming redundant or replaced by improved alternatives. To ensure our users have enough time to transition to the new endpoints, we follow a formal process. Initially, we mark any unnecessary endpoint as "deprecated". Subsequently, the endpoint is then scheduled for removal, a stage we refer to as "sunset". Finally, the endpoint is removed. This systematic approach allows for smooth transition and uninterrupted use of our API.

4.4.1. Deprecation

An endpoint that is deprecated is no longer receiving any updates, but still remains fully functional in the API.

Timeline for deprecation

An endpoint can get deprecated in a major release (T1–T3).

The deprecated endpoints are listed in the release notes.

HTTP header

Using a deprecated endpoint returns the following HTTP header in the response:

```
Deprecation: true
```

In addition, the "Link" header of the HTTP response has a link to this documentation page:

```
Link: <http://docs.cadmatic.org>; rel="deprecation", type=text/html
```

Note: The link header might also contain entries that are not related to the deprecation status.

4.4.2. Sunsetting

An endpoint that is sunset has a definite date after which the endpoint will be removed from the API. Until then, the endpoint is no longer receiving any updates, but still remains fully functional in the API.

For the users, the sunset date is not a fixed deadline: the endpoint keeps working until the user organization upgrades the Web API to a version that has been released after the sunset date.

Timeline for sunsetting

An endpoint can get sunset in the yearly T1 release, after being deprecated for some time.

The sunset endpoints are listed in the release notes.

HTTP header

Using a sunset endpoint returns the following kind of HTTP header in the response, in accordance with RFC 8594:

```
Sunset: HTTP-date
```

HTTP-date is defined in section-7.1.1.1 of RFC 7231. A real-world example of the response would look like this:

```
Sunset: Sat, 31 Dec 2025 23:59:59 GMT
```

In addition, the "Link" header of the HTTP response has a link to this documentation page:

```
Link: <http://docs.cadmatic.org>; rel="sunset", type=text/html
```

Note: The link header might also contain entries that are not related to the sunsetting status.

4.4.3. Removal

An endpoint that has been removed provides an error response if used.

Timeline for removal

An endpoint can get removed in the yearly T1 release, after being sunset for one year.

The removed endpoints are listed in the release notes.

Related Topics

Checking for deprecation or sunsetting

5. CADMATIC Web API Code Examples

The following examples demonstrate the use of CADMATIC Web API functions. They are either Python or Windows PowerShell scripts. The integration software that connects the 3rd party system and the CADMATIC Web API can be written in any programming language.

5.1. Creating access tokens

You must create an access token for a client that needs to access the endpoints of a given Web API service. Access tokens are only valid for 30 minutes; if you restart the Web API, existing tokens become invalid at once. If token expires, you must create a new one.

In actual integration, building access tokens must be part of the integration software. For testing purposes, you can build them with any suitable software, such as Windows PowerShell or Insomnia.

Prerequisites

- Name, Client ID, and Secret Key of the client. These are defined when creating a new client, as described in Adding clients.

Code example

The following Windows PowerShell code returns an access token. In the fields `URL`, `Name`, `ClientID`, and `ClientSecret`, replace the values with values from your project.

```
$Url = "http://localhost:25001/api/token"
$Body = @{
Name = "Client_1"
ClientID = "4A21D8D846BF98541184BEB96802BD9D"
ClientSecret = "irjdgoE9Dqa4KdAladb05zSB7JOo8IHQWO8NwAhqxz8="
}   | ConvertTo-Json

Invoke-RestMethod -Method 'Post' -Uri $Url -Body $Body -ContentType "application/json"
```

An example access token returned by the script:

```
token
-----

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NzY2NjgxNDN9.XyxYgHqj
n8Q2Moz2B3dANixII-7LpDbabhpTywHEDow
```

5.2. Authenticating clients

You can use one of the Authorization endpoints of the CADMATIC Web API to create an access token for your client. To request the token, you need to provide the Client ID and the Secret Key of your client. After this, the other endpoints will work if you pass the access token in the header of your REST requests. If you are using the Web API via Swagger UI, you do not need to add the token to the header: the Swagger page does that for you.

Prerequisites

- You have created a client for the Web API service.
- You know the Client ID and the Secret Key of your client.

Code example

The following Python code uses an access token to create an authenticated session, and then reads the list of available projects from the Web API.

```python
# Example how to authenticate to the Web API.
# This script will use the client ID and the client secret to authenticate to
# the Web API and receive an authentication token that can be used in
# subsequent requests. In this example, it is used to read the list of known
# project.
#
# Run this script with the following parameters:
# python webapi_authentication.py <Web API URL> <Client ID> <Client secret>

from requests import Session, post
from urllib.parse import urljoin

import sys
base_url, client_id, client_secret = sys.argv[1:4]

def create_session() -> Session:
    """Create an authenticated session to use with the Web API
    """
    token_url= urljoin(base_url, "api/token")
    reply = post(token_url, json={
        "ClientID": client_id,
        "ClientSecret": client_secret})
    if reply.status_code != 200:
        raise Exception("Authentication failure")
    token = reply.json()["token"]
    session = Session()
    session.headers.update({'Authorization': 'Bearer '+token})
    return session

with create_session() as session:
    projects = session.get(urljoin(base_url, "api/servers/projects")).json()
    print("Projects known to the Web API: ", projects)

```

5.3. Waiting for connections to projects

The CADMATIC Web API uses internal connectors to communicate with the projects it is connected to. After the Web API service has started, these internal connectors may still be starting, so trying to use them may result in a failure because the project is not connected, yet. You can use the */status* endpoint to check if a project is connected before sending requests to it.

Code example

The following Python code demonstrates how to connect to the Web API, get the list of all available projects, and wait until all the projects are ready to be used. Real client code would only need to wait for those projects that the client wants to use.

```python
# Example how the /status endpoints can be used.
# This script will authenticate to the Web API and poll the connection status
# of the hosted projects until all projects are connected.
#
# Run this script with the following parameters:
# python webapi_authentication.py <Web API URL> <Client ID> <Client secret>

from typing import List
from time import sleep
from requests import Session, post
from urllib.parse import urljoin

import sys
base_url, client_id, client_secret = sys.argv[1:4]

def create_session() -> Session:
    """Create an authenticated session to use with the Web API
    """
    token_url= urljoin(base_url, "api/token")
    reply = post(token_url, json={
        "ClientID": client_id,
        "ClientSecret": client_secret})
    if reply.status_code != 200:
        raise Exception("Authentication failure")
    token = reply.json()["token"]
    session = Session()
    session.headers.update({'Authorization': 'Bearer '+token})
    return session

def get_project_ids(s: Session) -> List[str]:
    """Get all project IDs known to the Web API
    """
    projects_url = urljoin(base_url, "api/servers/projects")
    projects = s.get(projects_url).json()
    return [p["project"]+"@"+p["serverOid"] for p in projects]

def check_and_print_connection_status(s: Session, ids: List[str]) -> bool:
    """Check and print the connection status of all connected projects
    """
    any_starting = False
    print("Connection status:")
    for id in ids:
        status_url = urljoin(base_url, "api/status/projects/"+id)
        status = s.get(status_url).json()["connectionStatus"]
        print(id+":\t"+status)
        any_starting = any_starting or (status == "Starting")
    return any_starting

with create_session() as session:
    project_ids = get_project_ids(session)
    while check_and_print_connection_status(session, project_ids):
        sleep(1)

```

5.4. Downloading files from publications

You can use the CADMATIC Web API to download files from document publications.

Prerequisites

- To run the example script, you need the OID of the target document. You can get it with *api/{projectName}/documents*.
- The script assumes that the target document has a publication with revision 'A'.

Code example

The following Windows PowerShell code gets the list of files from the publication and then downloads the files.

```
$ProjectDatabase = "ExampleProject.pms"
$DocumentName = "ExampleDrawing"
$DrawingRevision = "A"
$DrawingOid = "Ftoxvd5LGskntwQCM7bugW"

$token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1ODIxODM5MDV9.hDYjnQzoEi9xq2AXI1-lYJ5WtfQ6DWIwunkx0IoxSRU"

# Endpoint for listing files in a publication
$ListFilesUrl = "http://localhost:25000/api/$ProjectDatabase/documents/$DrawingOid/$DrawingRevision/files/"

$Headers = @{"Authorization" = "Bearer " + $token}

$filesList = Invoke-RestMethod -Method Get -Uri $ListFilesUrl -Headers $Headers -ContentType "application/json"

# Download each file with the file download endpoint
for (($i = 0); $i -le ($filesList.Length - 1); $i++)
    {
        $filename = $filesList[$i]
        $fileDownloadUrl = "http://localhost:25000/api/$ProjectDatabase/documents/$DrawingOid/$DrawingRevision/files/$filename"
        $savePath = "C:\tmp\" + $filename # Local place where we are saving the files

        Invoke-WebRequest -Method Get -Uri $fileDownloadUrl -Headers $Headers  -OutFile $savePath
    }
```

5.5. Creating EDM objects

You can use the CADMATIC Web API to create External Data Management (EDM) objects and assign attributes to them.

Prerequisites

- The attributes have been assigned to the "External Data Management" object type in COS. Otherwise, the system will display an error.

Code example

The following Windows PowerShell code creates a new EDM object and assigns the attribute "Description-Dsc" to it.

```powershell
ProjectDatabase = "ExampleProject.pms"
$token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1ODIxODYzNzJ9.DK9AxPlogjdmSEcTz-PsK2Iu7Tr97MM5SgCdCwgs4WE"
$Headers = @{"Authorization" = "Bearer " + $token}

# Define the EDM
$EDMs = @(
    @{
        positionId = "examplePositionID"
        externalId = "ID_2" # Must be unique among all EDM objects
        system = "rC8izVTyIbc9onnRefpwtG" # OID for a CADMATIC system
        line = "tyuEYX3qHl2ib096ieP.TG" # OID for a CADMATIC pipeline

        # Add one attribute, description. Must first be assigned to the EDM type
        attributes = @(
            @{
                tag = "Dsc"
                value = "Example EDM Object from Web API"
            }
        )
    }
)

# Convert the EDM to a JSON list, we need to specify depth = 3 since the attributes are nested inside the EDM object
$JSON_EDM_list = ConvertTo-Json -Depth 3 -InputObject $EDMs

# Create the EDM
$EDM_create_url = "http://localhost:25000/api/$ProjectDatabase/edm/create"

$result = Invoke-RestMethod -Method Post -Uri $EDM_create_url -Headers $Headers -Body $JSON_EDM_list -ContentType "application/json"

# Display the OID and external ID of the created object, they are returned as result
echo $result.oid
echo $result.externalId
```

5.6. Updating EDM objects

You can use the CADMATIC Web API to update an External Data Management (EDM) object with new attributes or data.

Note: If a Plant Modeller service is just checking out the EDM object for an update, for example because a user is inserting the object to the model at the same time, there may be a temporary error of not being able to update the EDM object.

Prerequisites

- To be able to update an EDM object, you need to know its OID. You get the OID when you create the EDM object.

Code example

The following Windows PowerShell code updates the value of the attribute "Description-Dsc" in an existing EDM object.

```powershell
$ProjectDatabase = "ExampleProject.pms"
$token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1ODIxODYzNzJ9.DK9AxPlogjdmSEcTz-PsK2Iu7Tr97MM5SgCdCwgs4WE"
$Headers = @{"Authorization" = "Bearer " + $token}

# Here we define the EDM we are updating, note that you must specify the OID
$EDMs = @(
    @{
        oid = "vovsaRG1GrwXxgToFQLRF0"
        positionId = "updated position ID"
        externalId = "ID_2" # Must be unique among all EDM objects
        system = "rC8izVTyIbc9onnRefpwtG" # OID for a CADMATIC system
        line = "tyuEYX3qHl2ib096ieP.TG" # OID for a CADMATIC pipeline

        # Add one attribute, description. Must first be assigned to the EDM type
        attributes = @(
            @{
                tag = "Dsc"
                value = "Updated description"
            }
        )
    }
)

# Convert the EDM to a JSON list, we need to specify depth = 3 since the attributes are nested inside the EDM object
$JSON_EDM_list = ConvertTo-Json -Depth 3 -InputObject $EDMs

# Now we can create the EDM
$EDM_update_url = "http://localhost:25000/api/$ProjectDatabase/edm/update"

$result = Invoke-RestMethod -Method Post -Uri $EDM_update_url -Headers $Headers -Body $JSON_EDM_list -ContentType "application/json"
```

5.7. Running model queries

You can use the CADMATIC Web API to run a model query.

Code example

The following Windows PowerShell code uses the endpoint that returns all attributes for all matching model objects. If you want to filter some of the attributes, use the function webapiModelQueryOid.

```
$ProjectDatabase = "ExampleProject.pms"
$token = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1ODIxODYzNzJ9.DK9AxPlogjdmSEcTz-PsK2Iu7Tr97MM5SgCdCwgs4WE"
$Headers = @{"Authorization" = "Bearer " + $token}

# OID of the model query is needed
$modelQueryOid = "qNGTv1DqGZo4jxgs2Kl0hW"

$modelQueryUrl = "http://localhost:25000/api/$ProjectDatabase/pm/model/query/all/$modelQueryOid"

# Result is a list of matching model objects. The OID of each object as well as their attributes are returned.
$modelObjects = Invoke-RestMethod -Method Get -Uri $modelQueryUrl -Headers $Headers -ContentType "application/json"

# Display the result in command prompt
echo $modelObjects
```

There is a Plant Modeller script for mass-creating Web API queries by selecting Systems with a user interface: *%PMS_HOME%\pm\macro\CreateWebApiQueriesBySystem.mac*

5.8. Checking for deprecation or sunsetting

Cadmatic follows a formal process of marking outdated endpoints first as deprecated and then as sunset, before finally removing them from the API. This process is described in more detail in Endpoint deprecation process.

Code example

The following Python code shows how to detect if a given endpoint is either deprecated or sunset by checking whether the HTTP header of the response includes either "Deprecation" or "Sunset". The same approach can be used with any endpoint in the Web API, to automate a pre-processing step.

```
from requests import Session, post

base_url = "http://localhost:25000"
client_id = "FF8036A4FA288DF28F121FBB26A6922F"
client_secret = "UHcIvsh7BUT023fXpu7tBdNo2NSJI4Wz3qwOgAM5xvI="
```

```python

def create_session() -> Session:

    """Create an authenticated session to use with the Web API
    """
    token_url = base_url+"/api/token"
    reply = post(token_url, json={
        "ClientID": client_id,
        "ClientSecret": client_secret})
    if reply.status_code != 200:
        raise Exception("Authentication failure")
    token = reply.json()["token"]
    session = Session()
    session.headers.update({'Authorization': 'Bearer '+token})
    return session

with create_session() as session:
    # Use the first configured project database
    projects = session.get(base_url+"/api/servers/projects").json()
    project_database = projects[0]["project"]

    parts = session.get(base_url+f"/api/{project_database}/parts/")
    if "Deprecation" in parts.headers.keys():
        print("Deprecated endpoint used")
    if "Sunset" in parts.headers.keys():
        print("Sunset endpoint used")
```