

Version

3.3



» Webservice Manual

September 2020

Author Tecnoteca srl

www.tecnoteca.com

ENG

www.cmdbuild.org

No part of this document may be reproduced, in whole or in part, without the express written permission of Tecnoteca s.r.l.

CMDBuild ® uses many great technologies from the open source community: PostgreSQL, Apache, Tomcat, Eclipse, Ext JS, JasperSoft, JasperStudio, Enhydra Shark, TWE, OCS Inventory, Liferay, Alfresco, GeoServer, OpenLayers, Quartz, BiMserver.
We are thankful for the great contributions that led to the creation of these products.

CMDBuild ® is a product of Tecnoteca S.r.l. which is responsible of software design and development, it's the official maintainer and has registered the CMDBuild logo.



CMDBuild ® is released under AGPL open source license (<http://www.gnu.org/licenses/agpl-3.0.html>)

CMDBuild ® is a registered trademark of Tecnoteca Srl.

Every time the CMDBuild® logo is used, the official maintainer "Tecnoteca srl" must be mentioned; in addition, there must be a link to the official website:

<http://www.cmdbuild.org>.

CMDBuild ® logo:

- cannot be modified (color, proportion, shape, font) in any way, and cannot be integrated into other logos
- cannot be used as a corporate logo, nor the company that uses it may appear as author / owner / maintainer of the project
- cannot be removed from the application, and in particular from the header at the top of each page

The official website is <http://www.cmdbuild.org>

Contents

| | |
|---|----|
| 1. Introduction..... | 6 |
| 1.1. The application..... | 6 |
| 1.2. Official website..... | 7 |
| 1.3. CMDBuild modules..... | 7 |
| 1.4. Available manuals..... | 7 |
| 1.5. Applications based on CMDBuild..... | 7 |
| 2. Interoperability standards..... | 9 |
| 2.1. Service-Oriented Architecture (SOA)..... | 9 |
| 3. Web services..... | 10 |
| 3.1. Web Service introduction..... | 10 |
| 3.2. SOAP Web Service introduction..... | 10 |
| 3.3. REST Web Service introduction..... | 11 |
| 4. SOAP Web Services..... | 13 |
| 4.1. CMDBuild WSDL..... | 13 |
| 4.2. SOAP Functions..... | 13 |
| 4.2.1. Cards..... | 13 |
| 4.2.2. Sessions..... | 14 |
| 4.2.3. Lookups..... | 14 |
| 4.2.4. Attributes..... | 15 |
| 4.2.5. Relations..... | 15 |
| 4.2.6. Classes..... | 16 |
| 4.2.7. Functions..... | 17 |
| 4.2.8. Attachments..... | 17 |
| 4.2.9. Reports..... | 17 |
| 4.2.10. Other functions..... | 18 |
| 5. REST Web Services..... | 19 |
| 5.1. CMDBuild REST web service..... | 19 |
| 5.2. CMDBuild REST Endpoints..... | 19 |
| 5.2.1. Async Operation..... | 20 |
| 5.2.2. Audits..... | 20 |
| 5.2.3. Bim project..... | 20 |
| 5.2.4. Bim values..... | 21 |
| 5.2.5. Boot..... | 21 |
| 5.2.6. Calendar Event Attachments..... | 22 |
| 5.2.7. Calendar Event Email..... | 23 |
| 5.2.8. Calendar Event..... | 23 |
| 5.2.9. Calendar Sequence..... | 24 |
| 5.2.10. Calendar Trigger..... | 26 |
| 5.2.11. Calendar View Event..... | 27 |
| 5.2.12. Card Attachments..... | 27 |
| 5.2.13. Card Bim values..... | 28 |
| 5.2.14. Card email attachments..... | 28 |
| 5.2.15. Card email..... | 29 |
| 5.2.16. Card geo values..... | 30 |
| 5.2.17. Card history..... | 31 |
| 5.2.18. Card locks..... | 31 |
| 5.2.19. Card print..... | 31 |
| 5.2.20. Card relations..... | 32 |
| 5.2.21. Cards..... | 32 |
| 5.2.22. Charset..... | 33 |
| 5.2.23. Class attributes..... | 33 |
| 5.2.24. Class filters..... | 35 |
| 5.2.25. Class or process domains..... | 36 |
| 5.2.26. Class print..... | 36 |

| | |
|---|----|
| 5.2.27. Class stats..... | 36 |
| 5.2.28. Classes..... | 37 |
| 5.2.29. Configurations..... | 38 |
| 5.2.30. Context menu component..... | 38 |
| 5.2.31. Custom pages..... | 39 |
| 5.2.32. Dashboard..... | 40 |
| 5.2.33. Dms category values..... | 40 |
| 5.2.34. Dms categories..... | 41 |
| 5.2.35. Dms models..... | 41 |
| 5.2.36. Domain attributes..... | 42 |
| 5.2.37. Domains..... | 43 |
| 5.2.38. Email accounts..... | 44 |
| 5.2.39. Email queue..... | 45 |
| 5.2.40. Email templates..... | 45 |
| 5.2.41. Etl Gate..... | 46 |
| 5.2.42. Etl templates..... | 47 |
| 5.2.43. Fk Domain..... | 49 |
| 5.2.44. Functions..... | 49 |
| 5.2.45. Geo attributes..... | 50 |
| 5.2.46. Geo style rules..... | 51 |
| 5.2.47. Geo values..... | 52 |
| 5.2.48. Geo server layers..... | 52 |
| 5.2.49. Grants..... | 53 |
| 5.2.50. Impersonation..... | 54 |
| 5.2.51. Jobs..... | 54 |
| 5.2.52. Language configurations..... | 55 |
| 5.2.53. Languages..... | 55 |
| 5.2.54. Locks..... | 56 |
| 5.2.55. Lookup types..... | 56 |
| 5.2.56. Lookup values..... | 56 |
| 5.2.57. Menu..... | 57 |
| 5.2.58. Minions..... | 58 |
| 5.2.59. Nav trees..... | 58 |
| 5.2.60. Process configuration..... | 59 |
| 5.2.61. Process instance activity email..... | 59 |
| 5.2.62. Process instance activity..... | 59 |
| 5.2.63. Process instance history..... | 60 |
| 5.2.64. Process instances..... | 60 |
| 5.2.65. Process start activities..... | 61 |
| 5.2.66. Process task definition..... | 61 |
| 5.2.67. Process task..... | 61 |
| 5.2.68. Processes..... | 61 |
| 5.2.69. Relations..... | 62 |
| 5.2.70. Reports..... | 63 |
| 5.2.71. Resources..... | 64 |
| 5.2.72. Role class filters..... | 64 |
| 5.2.73. Roles..... | 64 |
| 5.2.74. Session menu..... | 65 |
| 5.2.75. Session preferences..... | 65 |
| 5.2.76. Sessions..... | 66 |
| 5.2.77. System configuration..... | 66 |
| 5.2.78. System..... | 67 |
| 5.2.79. Tenants..... | 68 |
| 5.2.80. Timezones..... | 69 |
| 5.2.81. Translations..... | 69 |
| 5.2.82. Uploads..... | 70 |
| 5.2.83. Users..... | 70 |
| 5.2.84. Card views..... | 71 |
| 5.2.85. Views..... | 72 |
| 5.3. REST Examples..... | 73 |
| 5.3.1. Generating a session token..... | 73 |
| 5.3.2. Obtaining a list of every class..... | 74 |
| 5.3.3. Obtaining the information of a specific class..... | 75 |
| 5.3.4. Creating a new class..... | 76 |
| 5.3.5. Update an existing class..... | 77 |

| | |
|-----------------------------|----|
| 6. Appendix: Glossary..... | 79 |
| 6.1.1. ATTACHMENT..... | 79 |
| 6.1.2. WORKFLOW STEP..... | 79 |
| 6.1.3. ATTRIBUTE..... | 79 |
| 6.1.4. BIM..... | 79 |
| 6.1.5. CI..... | 79 |
| 6.1.6. CLASS..... | 80 |
| 6.1.7. CONFIGURATION..... | 80 |
| 6.1.8. DASHBOARD..... | 80 |
| 6.1.9. DATABASE..... | 80 |
| 6.1.10. DOMAIN..... | 80 |
| 6.1.11. DATA FILTER..... | 80 |
| 6.1.12. GIS..... | 81 |
| 6.1.13. GUI FRAMEWORK..... | 81 |
| 6.1.14. ITIL..... | 81 |
| 6.1.15. LOOKUP..... | 81 |
| 6.1.16. MOBILE..... | 81 |
| 6.1.17. PROCESS..... | 81 |
| 6.1.18. RELATION..... | 82 |
| 6.1.19. REPORT..... | 82 |
| 6.1.20. CARD..... | 82 |
| 6.1.21. SUPERCLASS..... | 82 |
| 6.1.22. ATTRIBUTE TYPE..... | 82 |
| 6.1.23. VIEW..... | 83 |
| 6.1.24. WEBSERVICE..... | 83 |
| 6.1.25. WIDGET..... | 83 |

1. Introduction

1.1. The application

CMDBuild is an open source web environment for the configuration of custom applications for the Asset Management.

On the one hand, it provides native mechanisms for the administrator, implemented in a "core" code which has been kept separated from the business logic, so that the system can be configured with all its features.

On the other hand, it generates dynamically a web interface for the operators, so that they can keep the asset situation under control and always know their composition, detachment, functional relations and how they update, in order to manage their life-cycle in a comprehensive way.

The system administrator can build and extend his/her own CMDB (hence the name of the project), modeling the CMDB according to the company needs; a proper interface allows you to progressively add new classes of items, new attributes and new relations. You can also define filters, "views" and access permissions limited to rows and columns of every class.

Using external visual editors, the administrator can design workflows, import them into CMDBuild and put them at operators' disposal, so that they can execute them according to the configured automatisms.

In a similar way, using external visual editors, the administrator can design various reports on CMDB data (printouts, graphs, barcode labels, etc.), import them into the system and put them at operators' disposal.

The administrator can also configure some dashboards made up of charts which immediately show the situation of some indicators in the current system (KPI).

A task manager included in the user interface of the Administration Module allows you to schedule various operations (process starts, e-mail receiving and sending, connector executions) and to control CMDB data (synchronous and asynchronous events). Based on their findings, it sends notifications, starts workflows and executes scripts.

Thanks to document management systems that support the CMIS standard (Content Management Interoperability Services) - among which there is also the open source solution Alfresco - you will be able to attach documents, pictures, videos and other files.

Moreover, you can use GIS features to georeference and display assets on a geographical map (external map services) and / or on vector maps (local GeoServer and spatial database PostGIS) and BIM features to view 3D models (IFC format).

The system also includes a REST webservice, so that CMDBuild users can implement custom interoperability solutions with external systems.

Furthermore, CMDBuild includes two external frameworks:

- the Advanced Connector CMDBuild, which is written in Java and can be configured in Groovy: it helps the implementation of connectors with external data sources, i.e automatic inventory systems, virtualization or monitoring ones (supplied with non-open source licence to the users that subscribe the annual Subscription with Tecnoteca)
- the GUI Framework CMDBuild, which helps the implementation of additional graphical interfaces, i.e. web pages (simplified for non technicians) that have to be published on external portals and that are able to interact with the CMDB through the REST webservice

CMDBuild includes a mobile interface (for smartphone and tablet). It is implemented as multi-platform app (iOS, Android) and is able to interact with the CMDB through the REST webservice (supplied with non-open source licence to the users that subscribe the annual Subscription with Tecnoteca).

CMDBuild is an enterprise system: server-side Java, web Ajax GUI, SOA architecture (Service Oriented Architecture), based on webservice and implemented by using the best open source technologies and following the sector standards.

CMDBuild is an ever-evolving system, which has been released for the first time in 2006 and updated several times a year in order to offer more features and to support new technologies.

1.2. Official website

CMDBuild has a dedicated website: <http://www.cmdbuild.org>

The website gathers a lot of documents on technical and functional features of the project: brochures, slides, manuals (see next paragraph), testimonials, case histories, newsletters, forums.

1.3. CMDBuild modules

The CMDBuild application includes two main modules:

- the Administration Module for the initial definition and the next changes of the data model and the base configuration (relation classes and typologies, users and authorization, dashboards, upload report and workflows, options and parameters)
- the Management Module, used to manage cards and relations, add attachments, run workflow processes, visualize dashboards and execute reports

The Administration Module is available only to the users with the "administrator" role; the Management Module is used by all the users who view and edit data.

1.4. Available manuals

This manual is dedicated to the Administration Module, through which the administrator can configure data, define users and permissions, and perform other tasks.

You can find all the manuals on the official website (<http://www.cmdbuild.org>):

- system overview ("Overview Manual")
- system administration ("Administrator Manual")
- installation and system management ("Technical Manual")
- workflow configuration ("Workflow Manual")
- webservice details and configuration ("Webservice Manual")
- connectors to sync data through external systems ("ConnectorsManual")

1.5. Applications based on CMDBuild

Tecnoteca has used the CMDBuild environment in order to implement two different pre-configured solutions:

- CMDBuild READY2USE, for the management of assets and IT services, oriented to internal IT infrastructures or services for external clients (<http://www.cmdbuild.org/it/prodotti/ready2use>) according to the ITIL best practice (Information Technology Infrastructure Library)
- openMAINT, for the inventory management of assets, properties and related maintenance

activities (<http://www.openmaint.org>)

Both applications are released with open source license, except for certain external components (data sync connectors, Self-Service portal, mobile APP, etc.), that are reserved to the users that subscribe the annual Subscription with Tecnoteca.

2. Interoperability standards

2.1. Service-Oriented Architecture (SOA)

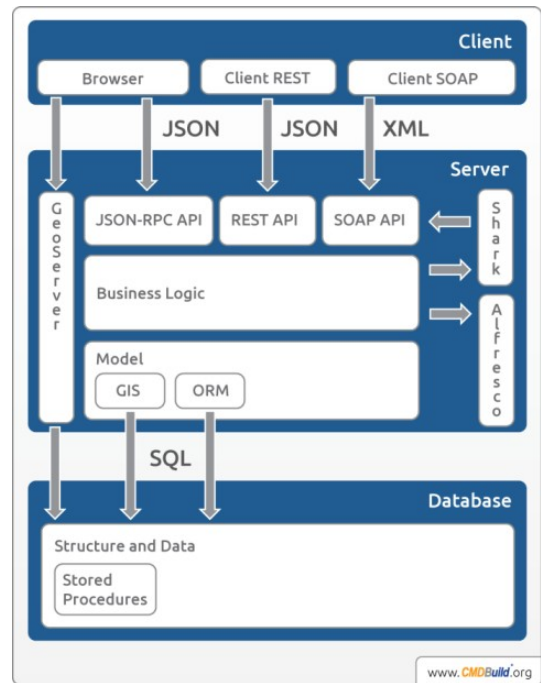
In order to make different applications interoperable, they must be created as components that cooperate with the services implementation, and these services must be set through high level interfaces defined under standard protocols.

CMDBuild is designed with Service-Oriented Architecture (SOA):

- decoupling the different logic levels (see the schema)
- implementing and setting in every interface external specifications as a single modality for the access to relating data and methods
- using the interfaces both for the interactive access of the web client and for the programmatic access of external applications

From a technical point of view, we chose to use the following technology of web services:

1. REST protocol
2. SOAP protocol



Through web services, and safety policy permitting, CMDBuild provides the data filed in the CMDB and its management methods to allow the use within other applications involved with the information itself, both for the technical management and for administration.

3. Web services

3.1. Web Service introduction

A web service is an interface that describes a collection of methods, available over a network and working using XML messages.

With web services, an application allows other applications to interact with its methods.

Nowadays the two most used standards are:

- SOAP Web Services
- REST Web Services

In the following chapters both standards will be introduced, with a list of their differences and some examples.

3.2. SOAP Web Service introduction

SOAP (Simple Object Access Protocol) is a protocol based on XML language. Thanks to the XML usage SOAP, unlike other frameworks, provides a platform and language independent communication.

The structure of SOAP messages is divided in four parts:

- Envelope: a mandatory element that defines the beginning and the end of the message;
- Header: an optional element that contains any optional attributes;
- Body: a mandatory element that provides the message that has to be sent;
- Fault: a mandatory element that can provide any error that occurs while processing the message;

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="http://soap.services.cmdbuild.org">
  <soapenv:Header>
    ...
  </soapenv:Header/>
  <soapenv:Body>
    <soapenv:Fault>
      ...
    </soapenv:Fault>
    ...
  </soapenv:Body>
</soapenv:Envelope>
```

In the usage of SOAP for CMDBuild the header will be used mainly to authenticate the user, by adding a security field where the user can provide his username and password to access the service.

In the body field the user can provide the function that has to be called with the following syntax:

```
<soap:functionName/>
```

As an example of usage in CMDBuild, the following SOAP request will generate a session for the user specified in the header username field:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap="http://soap.services.cmdbuild.org">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>username</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-username-token-profile-1.0#PasswordText">password</wsse:Password>
      </wsse:UsernameToken></wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
      <soap:createSession/>
    </soapenv:Body>
  </soapenv:Envelope>
```

When executing a SOAP request, the target endpoint, in the CMDBuild case, has to be the Private.wsdl file in the CMDBuild installation. The WSDL file provides the user with a list of all available functions that can be called.

For a more detailed description on what you can do with SOAP on CMDBuild read chapter 4.

3.3. REST Web Service introduction

REST (REpresentational State Transfer), unlike SOAP, is an architectural style that provides a stateless, simple and lightweight way of communicating with a system.

The format used to send and receive data with REST web services is JSON. This format is a simple text containing a series of attribute-value pairs, like the following:

```
{
  "Key1": "value1",
  "Key2": "value2",
  ...
}
```

When REST is used, requests can be sent to various endpoints through GET, PUT, POST and DELETE HTTP requests.

Elements such as authentication tokens can be added in the header of the request, data that has to be sent through PUT or POST requests can be added as parameters of the request.

As an example, if we want to generate a session like we previously did with the SOAP web service, we would firstly need to get the session endpoint of CMDBuild:

`http://hostname:port/cmdbuild/services/rest/v3/sessions`

And then perform a POST request with the username and password of the user to authenticate.

The response would then contain a success key followed by a data key containing the response values (from version 3.2 due to security reasons an additional request parameter has to be set to true in order to obtain the sessionId in the response, more at chapter 5.3.1):

```
{
  "success": "true",
  "data": [ "sessionId": "generatedSessionId",
           "username": "user",
           "password": "userPassword" ]
}
```

For a more detailed description on what you can do with REST on CMDBuild read chapter 5.

4. SOAP Web Services

4.1. CMDBuild WSDL

To obtain a list of every possible function that can be called through SOAP web services CMDBuild provides a WSDL called Private.wsdl. It can be opened with a normal text editor to visualize an XML containing the definition of every element, but it can also be opened by a software like SoapUI to have a more clear view of its content. To access this file once CMDBuild is up and running you can use the following URL: <http://host:port/cmdbuild/services/soap/Private?wsdl>

In the next paragraph a list of every available function will be provided.

4.2. SOAP Functions

In the following table a list of available SOAP functions, divided by category, will be provided, note that after future updates the functions might change, so it's always better to verify the function format in the WSDL file.

4.2.1. Cards

Card data structure:

- `className`: a string that identifies the owner class
- `id`: a bigint to identify the card
- `attributeList`: an array of attributes of the card
- `beginDate`: a date that shows the creation date of the card
- `user`: a string that shows what user last modified the card

| Function | Parameters | Description |
|-----------------------------|--|--|
| <code>getCard</code> | <code>-className</code> <code>-cardId</code> <code>-attributeList</code> | Function used to obtain the information relative to a specific card owned by a specific class |
| <code>getCardHistory</code> | <code>-className</code> <code>-cardId</code> <code>-limit</code> <code>-offset</code> | Function used to obtain the history of a specific card owned by a specific class. With limit and offset the amount of results can be modified |
| <code>getCardList</code> | <code>-className</code> <code>-attributeList</code> <code>-queryType</code> <code>-orderType</code> <code>-limit</code> <code>-offset</code> <code>-fullTextQuery</code> <code>-cqlQuery</code> | Function used to obtain a full list of cards owned by a specific class, with the possibility of specifying a filter (with Query type, FullTextQuery or CqlQuery), with the possibility of controlling the number and order of results via Limit, Offset and Order list |
| <code>getCardListExt</code> | <code>-className</code> <code>-attributeList</code> <code>-queryType</code> | Function used to obtain a full extended list of cards owned by a specific class, with the possibility of specifying a filter (with Query type, |

| | | |
|-------------------------------|--|--|
| | -orderType -limit -offset -fullTextQuery -cqlQuery | FullTextQuery or CqlQuery), with the possibility of controlling the number and order of results via Limit, Offset and Order list |
| getCardListWithLongDateFormat | -className -attributeList -queryType -orderType -limit -offset -fullTextQuery -cqlQuery | Function used to obtain a full list of cards owned by a specific class with a long date format, with the possibility of specifying a filter (with Query type, FullTextQuery or CqlQuery), with the possibility of controlling the number and order of results via Limit, Offset and Order list |
| getCardMenuSchema | | |
| createCard | -className -attributeList -beginDate -endDate -metadata | Create a card owned by the specified class with a list of attributes specified in the request |
| updateCard | -className -attributeList -beginDate -endDate -id -metadata | Update a card owned by the specified class with the values specified in the request |
| deleteCard | -className -cardId | Delete a specific card owned by a specific class |

4.2.2. Sessions

| Function | Parameters | Description |
|---------------|------------|--|
| createSession | | Create a session for the user with username and password specified in the header |

4.2.3. Lookups

Lookup data structure:

- id: a bigint to identify the lookup
- type: a string to identify the name of the lookup list which includes the current heading
- description; a string to describe the lookup heading
- code
- parent: the parent of the current lookup
- parentId: the id of the parent of the current lookup
- position: the position of the lookup in the lookup list
- notes: a string containing the optional notes of the lookup

| Function | Parameters | Description |
|-------------------|--|---|
| getLookupById | -id | Get a specific lookup by specifying its Id |
| getLookList | | Get the full list of lookups |
| getLookListByCode | -type -code | Get the list of lookups of a specific type with a specific code |
| createLookup | -code -description -notes -parent -type | Create a new lookup with the values defined in the parameters. The parent parameters requires a parentId and a position |
| updateLookup | -code -description -id -notes -parent -type | Update a lookup with the id specified in the parameters with the values |
| deleteLookup | -id | Delete the lookup with the specified Id |

4.2.4. Attributes

Attribute data structure:

- name: a string that defines the attribute name
- value: a string to identify the attribute value
- code

| Function | Parameters | Description |
|------------------|------------|---|
| getAttributeList | -className | Get a list of attributes of the class specified with the class name parameter |

4.2.5. Relations

Relation data structure:

- domainName: a string that defines the domain used for the relation
- class1Name: a string to identify the first class of the relation
- card1Id: a bigint to identify the first card of the relation
- class2Name: a string to identify the second class of the relation
- card2Id: a bigint to identify the second card of the relation

| Function | Parameters | Description |
|-----------------------|---|---|
| getRelationAttributes | -class1Name -class2Name -card1Id -card2Id -domainName | Get the attributes of a specific relation |

| | | |
|------------------------------|--|--|
| getRelationHistory | -class1Name -class2Name -card1Id -card2Id -domainName | Get the history of a specific relation |
| getRelationList | -className -cardId | Get the full list of relations of a specific card owned by the class specified by Class name |
| getRelationListExt | -domain -className -cardId | Get the full extended list of relations of a specific card owned by the class specified by Class name |
| createRelation | -class1Name -class2Name -card1Id -card2Id -domainName | Create a relation between the two classes and two cards specified in the parameters |
| createRelationWithAttributes | -class1Name -class2Name -card1Id -card2Id -domainName -attributes | Create a relation between the two classes and two cards specified in the parameters with the provided list of attributes |
| updateRelationAttributes | -class1Name -class2Name -card1Id -card2Id -domainName -attributes | Update the attributes owned by the relation between the specified cards and classes provided in the parameters |
| deleteRelation | -class1Name -class2Name -card1Id -card2Id -domainName | Delete the specified relation |

4.2.6. Classes

| Function | Parameters | Description |
|----------------------|--------------------------------|---|
| getClassSchema | -className | Get the schema of a the class with the name specified in the parameters |
| getClassSchemaById | -classId -includeAttributes | Get the schema of a the class with the id specified in the parameters and, optionally, include its attributes |
| getClassSchemaByName | -className | Get the schema of a the class with the name specified in the parameters |

4.2.7. Functions

| Function | Parameters | Description |
|-----------------|---|--|
| callFunction | -functionName -code -name -value | Execute the specified function |
| getFunctionList | | Obtain a list of all available functions |

4.2.8. Attachments

Attachment data structure:

- category: a string that identifies the category of the attribute
- description: a string that represent the description of the attachment
- filename: a string that contains the name of the file with the extension
- version: a string containing the version of the attachment
- author: a string containing the author of the upload
- created: a date indicating when the file was firstly uploaed
- modified: a date indicating when the file was last modifies

| Function | Parameters | Description |
|-----------------------------|--|--|
| copyAttachment | -sourceClassName -sourceId -destinationClassName -destinationId | Copy an attachment from one class to another class |
| updateAttachmentDescription | -className -filename -description | Modify the description of the attachment with name Filename owned by the specified class |
| downloadAttachment | -className -objectId -filename | Download the specified attachment |

4.2.9. Reports

| Function | Parameters | Description |
|------------------|------------------------------|--|
| getBuiltInReport | -id -extension -params | Get a report with the specified id and extension |
| getReport | -id -extension -params | Get a report with the specified id and extension |
| getReportList | -type -limit -offset | Get a full list of reports of the specified type |

| | | |
|---------------------|-------------------|--|
| getReportParameters | -id -extension | Obtain a list of all report parameters for the specified report with the specified extension |
|---------------------|-------------------|--|

4.2.10. Other functions

| Function | Parameters | Description |
|-----------------------|---|--|
| abortWorkflow | -card | Abort a specific workflow for the specified card in the parameters |
| generateDigest | -plainText -digestAlgorithm | Generate a digest with the specified algorithm |
| getActivityMenuSchema | | Get the activity menu schema |
| getActivityObjects | -className -cardId | Get a list of activity objects for a specific card owned by the class with className |
| getMenuSchema | | Get the menu schema |
| getProcessHelp | -className -cardId | Get the process help for the specified card owned by the class with name className |
| getReference | -className -query -orderType -limit -offset -fullTextQuery -cqlQuery - | Get the specified reference |
| getUserInfo | | Get the information about the authenticated user |
| notify | | |
| resumeWorkflow | -card | Resume the workflow of a card |
| suspendWorkflow | -card | Suspend the workflow of a card |
| updateWorkflow | -card | Update the workflow of a card |
| sync | -xml | |

5. REST Web Services

5.1. CMDBuild REST web service

Unlike SOAP, for the REST webservices there isn't a file containing every function that can be requested on the same endpoint, but there are specific endpoints for every action.

In example if we want to operate on a card the endpoint will be:

`http://hostname:port/cmdbuild/services/rest/v3/cards`

And with GET, POST, PUT, DELETE requests and addition to the endpoint path we can get a list of cards or update a specific card.

5.2. CMDBuild REST Endpoints

In this paragraph a list of currently available endpoints and data structure will be presented.

Note that in some cases the endpoint can have some variations in the path, for example some endpoints are the same for cards or process instances, and when that occurs the “[” symbol is used.

When an endpoint requires additional information in the path it will be specified in the Path column.

When an endpoint requires additional query params, those will be specified in the Parameters column.

A data structure is presented whenever a POST/PUT endpoint requires a custom data structure containing the required details in a json format;

The majority of endpoints will provide a function to read the entire list of objects, a function to get the details of a specific object, a function to create a new object, a function to update an existing object and a function to delete an existing object.

In the path of various endpoints, whenever there is “id”, “classId”, “cardId” or similar, that has to be substituted with the id (or code when talking about classes) of the required element.

Before version 3.3 many endpoints that managed lists of items had the possibility of specifying the filter, limit, sort and other constraints to the output. To avoid listing every time all the different parameters a new object called `WsQueryOptions` has been created to always include those parameters. When the parameter “`wsQueryOptions`” is specified for an endpoint it might be possible to use the followings:

- `attrs` List of attributes to return
- `filterStr` A string containing a filter
- `sort` A string containing the sorting function
- `limit` A long value to limit the amount of results
- `start` A long value to set an offset in the resultset
- `detailed` A boolean to force the server to return a detailed response
- `positionOf` A long value containing an id to return in the meta field

| | | | | |
|-----------------------|----------|--|--------|---|
| getOne | /id | | GET | Obtain the specified Bim project |
| createProjectWithFile | | -data json -file multipart dataHandler -ifcFormat String | POST | Create a new Bim project |
| update | /id | -data json | PUT | Update the specified Bim project |
| downloadIfcFile | /id/file | -ifcFormat String | GET | Download the specified Ifc file for the specified Bim project |
| uploadIfcFile | /id/file | -file dataHandler -ifcFormat String | POST | Upload a new Ifc file for the specified Bim project |
| delete | /id | | DELETE | Delete the specified bim project |

5.2.4. Bim values

<http://hostname:port/cmdbuild/services/rest/v3/bim/values>

| Function | Path | Parameters | Type | Description |
|----------|-----------|--------------------|------|--|
| getOne | /globalId | -if_exists Boolean | GET | Obtain the Bim values of the specified Bim project |

5.2.5. Boot

<http://hostname:port/cmdbuild/services/rest/v3/boot>

| Function | Path | Parameters | Type | Description |
|---------------------|---------------------|---|------|--|
| status | /status | | GET | Get the current status of the system |
| checkDatabaseConfig | /database/check | -dbConfig Map<String,String> | POST | |
| reconfigureDatabase | /database/configure | -file multipart dataHandler -dbConfig Map<String,String> | POST | |
| getPendingPatches | /patches | | GET | Obtain a list of currently available patches |

| | | | | |
|---------------------|----------------|--|------|--|
| applyPendingPatches | /patches/apply | | POST | Tell the system to perform the installation of every currently available patch |
|---------------------|----------------|--|------|--|

5.2.6. Calendar Event Attachments

http://hostname:port/cmdbuild/services/rest/v3/calendar/events/eventId/attachments

Attachment data structure:

- category String
- fileName String
- majorVersion boolean
- cardValues Map<String, Objects>

| Function | Path | Parameters | Type | Description |
|-------------------------|------------------------------------|--|--------|---|
| create | | -attachment attachmentData -file DataHandler -copyFrom_class String -copyFrom_card Long -copyFrom_id String | POST | Create a new attachment for the specified calendar event with the possibility of copying the attachment from another card |
| readMany | | -wsQueryOption | GET | Obtain a list of all the attachments for the specified event |
| readOne | /attachmentId | | GET | Obtain a single attachment with the specified attachmentId |
| download | /attachmentId/download | | GET | Download the file of a specific attachment |
| preview | /attachmentId/preview | | GET | Get the attachment preview if available |
| update | /attachmentId | -attachment attachmentData -file DataHandler | PUT | Update a specific attachment with the provided data |
| delete | /attachmentId | | DELETE | Delete an attachment with the specified id |
| getAttachmentHistory | /attachmentId/history | | GET | Obtain the history of a specific attachment |
| downloadPreviousVersion | /attachmentId/history/version/file | | GET | Get an older version of a specific attachment |

5.2.7. Calendar Event Email

<http://hostname:port/cmdbuild/services/rest/v3/calendar/events/eventId/emails>

| Function | Path | Parameters | Type | Description |
|----------|----------|--|--------|---|
| create | | -applyTemplate Boolean -template_only Boolean -attachments List<String> -parts List<Attachment> | POST | Create a new email for the specified event with the possibility of adding a list of attachmentIds |
| update | /emailId | -emailData wsEmailData | PUT | Update an existing email |
| readAll | | -wsQueryOptions | GET | Obtain a list of every email associated with the specified event |
| read | /emailId | | GET | Get the details of a specific email related to a specific event |
| delete | /emailId | | DELETE | Delete a sepcific email |

5.2.8. Calendar Event

<http://hostname:port/cmdbuild/services/rest/v3/calendar/events>

Event data structure:

- category String
- priority String
- job Long
- card Long
- sequence Long
- content String
- description String
- timeZone String
- eventEditMode String
- notifications List<String>
- partecipants List<String>
- onCardDeleteAction String
- type String
- begin String
- end String

- owner String
- status String
- source String
- notes String

| Function | Path | Parameters | Type | Description |
|------------------|-------------------------------|---|--------|--|
| readMany | | -filter String -sort String -limit Long -start Long -detailed Boolean -positionOf Long -goToPage Boolean | GET | Obtain a list of every event |
| readOne | /eventId | -includeStats Boolean | GET | Get the details of a specific event |
| createUserEvent | | -data json | POST | Create a new event with the provided data |
| update | /eventId | -data json | PUT | Update an existing event with the provided data |
| delete | /eventId | | DELETE | Remove an existing event |
| readHistory | /eventId/history | -limit Long -start Long -detailed Boolean | GET | Obtain the history of a specific event |
| getHistoryRecord | /eventId/history/ recordId | | GET | Obtain the details of a specific record from the history of an event |

5.2.9. Calendar Sequence

<http://hostname:port/cmdbuild/services/rest/v3/calendar/sequences>

Sequence data structure:

- category String
- priority String
- job Long
- card Long

- content String
- description String
- timeZone String
- title String
- eventCount Integer
- frequencyMultiplier Integer
- maxActiveEvents Integer
- eventEditMode String
- notifications List<String>
- participants List<String>
- onCardDeleteAction String
- sequenceParamsEditMode String
- showGeneratedEventsPreview Boolean
- eventType String
- firstEvent String
- lastEvent String
- trigger Long
- endType String
- events List<WsEventData>

| Function | Path | Parameters | Type | Description |
|------------------|-----------------------|---|--------|---|
| readOne | /sequenceId | -includeEvents Boolean | GET | Obtain the details of a specific sequence, with the possibility of including the related events |
| readManyByCard | /by-card/cardId | -detailed Boolean -includeEvents Boolean | GET | Obtain a list of sequences related to a specific card |
| create | | -data json | POST | Create a new sequence with the provided data |
| update | /sequenceId | -data json | PUT | Update an existing sequence with the provided data |
| delete | /sequenceId | | DELETE | Remove a specific sequence |
| getEventsPreview | /_ANY/generate-events | -data json | POST | Generate events based on the provided data |

5.2.10. Calendar Trigger

http://hostname:port/cmdbuild/services/rest/v3/calendar/triggers

Trigger data structure:

- category String
- priority String
- job Long
- conditionScript String
- content String
- description String
- timeZone String
- eventCount Integer
- frequencyMultiplier Integer
- maxActiveEvents Integer
- delay String
- eventEditMode String
- eventTime String
- frequency String
- notifications List<String>
- participants List<String>
- onCardDeleteAction String
- sequenceParamsEditMode String
- showGeneratedEventsPreview Boolean
- active Boolean
- eventType String
- ownerClass String
- ownerAttr String
- endType String

| Function | Path | Parameters | Type | Description |
|--------------------|------------------------------|----------------------|------|--|
| readOne | /triggerId | | GET | Obtain the details of a specific trigger |
| getSequencePreview | /triggerId/generate-sequence | -dateValue String | GET | Generate a sequence based on the provided data |
| createEvents | /triggerId/create-events | -WsQueryOptions | POST | Force the event creation for the specified trigger |

| | | | | |
|----------|------------|------------|--------|---|
| readMany | | | GET | Get a full list of calendar triggers |
| create | | -data json | POST | Create a new trigger with the provided data |
| update | /triggerId | -data json | PUT | Update an existing trigger with the provided data |
| delete | /triggerId | | DELETE | Delete an existing trigger |

5.2.11. Calendar View Event

<http://hostname:port/cmdbuild/services/rest/v3/calendar/views/viewId/events>

| Function | Path | Parameters | Type | Description |
|----------|----------|---|------|---|
| readOne | /eventId | | GET | Get the details of a specific event of a view |
| readMany | | -limit Long -start Long -detailed Boolean | GET | Obtain a list of events of a specific view |

5.2.12. Card Attachments

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/cards|instances/cardId/attachments>

| Function | Path | Parameters | Type | Description |
|--------------|------------------------|---|------|---|
| create | | -attachment attachmentData -file DataHandler -copyFrom_class String -copyFrom_card Long -copyFrom_id String | POST | Create a new attachment for the specified card with the possibility of copying the attachment from another card |
| readMany | | -wsQueryOption | GET | Obtain a list of all the attachments for the specified card |
| readOne | /attachmentId | | GET | Obtain a single attachment with the specified attachmentId |
| download | /attachmentId/download | | GET | Download the file of a specific attachment |
| downloadMany | /_MANY/file | -attachmentId List<String> | GET | Obtain many attachments file based on the id list |

| | | | | |
|-------------------------|---|---|--------|---|
| download | /attachmentId/ file | | GET | Download the attachment with id attachmentId of the specified email |
| preview | /attachmentId/ preview | | GET | Obtain a preview of the attachment with id attachmentId of the specified email |
| update | /attachmentId | -attachment multipart json -file multipart dataHandler | PUT | Update an existing attachment with id attachmentId of the specified email with the given data |
| delete | /attachmentId | | DELETE | Delete a specific attachment with if attachmentId of the email with id emailId |
| getAttachmentHistory | /attachmentId/ history | | GET | Obtain the history of a specific attachment |
| downloadPreviousVersion | /attachmentId/ history/version/ file: | | GET | Download the specified previous version of the attachment with id attachmentId of the email with if emailId |

5.2.15. Card email

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/cards|instances/cardId/emails>

Email data structure:

- delay Long
- from String
- replyTo String
- to String
- cc String
- bcc String
- subject String
- body String
- contentType String
- account Long
- template Long
- autoReplyTemplate Long
- keepSynchronization boolean

- noSubjectPrefix boolean
- promptSynchronization boolean
- status String
- _expr String

| Function | Path | Parameters | Type | Description |
|----------|----------|--|--------|--|
| readAll | | -limit Integer -start Integer -detailed Boolean | GET | Obtain a list of all available email associated with a card, with the possibility of changing the response to detailed or not and the possibility of change the number of results with limit |
| read | /emailId | | GET | Obtain the details of a specific email |
| create | | -data List<json> -apply_template Boolean -template_only Boolean | POST | Create an email associated with a card with the data provided in emailData, with the possibility of extending a pre-existing template or using only a template |
| update | /emailId | -data json -apply_template Boolean -template_Only Boolean | PUT | Update an existing email associated with a card with the provided data in emailData |
| delete | /emailId | | DELETE | Delete a specific email |

5.2.16. Card geo values

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/cards|instances/cardId/geovalues>

Geo values data structure:

- _type String
- x Double
- y Double
- points List<WsPoint>

Point data structure:

- x Double
- y Double

| Function | Path | Parameters | Type | Description |
|---------------|--------------|---------------|--------|--|
| getAllForCard | | | GET | Obtain every geovalue associated with a card |
| get | /attributeId | | GET | Obtain the details of a specific geovalue |
| set | /attributeId | -data json | PUT | Update an existing geovalue with the provided data |
| delete | /attributeId | | DELETE | Delete a specific geovalue |

5.2.17. Card history

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/cards|instances/cardId/history>

| Function | Path | Parameters | Type | Description |
|------------------|-----------|----------------------------------|------|--|
| getHistory | | -limit Long -start Long | GET | Get the history of a specific card |
| getHistoryRecord | /recordId | | GET | Get the details of a specific card in the card history |

5.2.18. Card locks

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/cards|instances/cardId/lock>

| Function | Path | Parameters | Type | Description |
|-------------|------|------------|--------|------------------------------------|
| getLock | | | GET | Get the lock of a specific card |
| createLock | | | POST | Create a lock for a specific card |
| releaseLock | | | DELETE | Release a lock for a specific card |

5.2.19. Card print

[http://hostname:port/cmdbuild/services/rest/v3/classes/classId/cards/cardId/print/file:](http://hostname:port/cmdbuild/services/rest/v3/classes/classId/cards/cardId/print/file)

| Function | Path | Parameters | Type | Description |
|----------|------|----------------------|------|---|
| readOne | | -extension String | GET | Download a file with an arbitrary extension, with the details of a specific card with cardId, owned by a class with classId |

5.2.20. Card relations

http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/cards|instances/cardId/relations

Relation data structure:

- `_id` Long
- `_type` String
- `_sourceType` String
- `_sourceId` Long
- `_destinationType` String
- `_destinationId` Long
- `_is_direct` boolean

| Function | Path | Parameters | Type | Description |
|----------|-------------|--|--------|--|
| read | | -limit Long -start Long -detailed Boolean | GET | Get a list of relations for a specific card |
| create | | -data json | POST | Create a new relation for a specific card with the provided data |
| update | /relationId | -data json | PUT | Update an existing relation for a specific card with the provided data |
| delete | /relationId | | DELETE | Delete a specific relation |

5.2.21. Cards

http://hostname:port/cmdbuild/services/rest/v3/classes/classId/cards

Card data structure:

The data structure for a card is a map of string and objects that vary based on

| Function | Path | Parameters | Type | Description |
|----------|---------|--|------|---|
| readOne | /cardId | -includeModel Boolean -includeWidgets Boolean -includeStats Boolean | GET | Obtain the details of a specific card |
| readMany | | WsQueryOptions -functionValue | GET | Get a list of all available cards owned by a class, |

| | | | | |
|------------|---------|--|--------|---|
| | | String -distinctIncludeNull Boolean -distinct String -count String | | with the possibility of adding filters to limit the results |
| create | | -data json | POST | Create a new card with the provided data for a specific class |
| update | /cardId | -data json | PUT | Update an existing card with the provided data |
| delete | /cardId | | DELETE | Delete a specific card |
| updateMany | | WsQueryOptions -data json | PUT | Update many cards at once |
| deleteMany | | WsQueryOptions | DELETE | Delete many cards at once |

5.2.22. Charset

<http://hostname:port/cmdbuild/services/rest/v3/system/charsets>

| Function | Path | Parameters | Type | Description |
|-----------------------|------|------------|------|---|
| readAvailableCharsets | | | GET | Obtain a list of all available charsets |

5.2.23. Class attributes

<http://hostname:port/cmdbuild/services/rest/v3/processes/classes/classId/attributes>

Attribute data structure:

- formatPattern String
- unitOfMeasure String
- unitOfMeasureLocation String
- visibleDecimals Integer
- preselectIfUnique boolean
- showThousandsSeparator boolean
- showSeconds boolean
- showSeparators boolean
- type String
- name String
- description String
- showInGrid boolean
- showInReducedGrid boolean
- domainKey String

- domain String
- direction String
- unique boolean
- mandatory boolean
- active boolean
- index Integer
- defaultValue String
- group String
- precision String
- scale String
- targetClass String
- maxLenght Integer
- editorType String
- lookupType String
- filter String
- help String
- showIf String
- validationRules String
- mode boolean
- autoValue String
- metadata map<String, String>
- classOrder Integer
- isMasterDetail Boolean
- masterDetailDescription String
- ipType String
- textContentSecurity String

| Function | Path | Parameters | Type | Description |
|----------|---------|----------------------------------|------|--|
| read | /attrId | | GET | Obtain the details of a specific attribute |
| readAll | | -limit Long -start Long | GET | Obtain a list of every available attribute for the specified class |
| create | | -data json | POST | Create a new attribute for a specific class with the provided data |

5.2.25. Class or process domains

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/domains>

| Function | Path | Parameters | Type | Description |
|------------|------|----------------------|------|---|
| getDomains | | -detailed Boolean | GET | Obtain a list of all domains related to a class or process, with the possibility of specifying if obtaining the full response or a basic response with the boolean parameter includeFullDetails |

5.2.26. Class print

<http://hostname:port/cmdbuild/services/rest/v3/classes>

| Function | Path | Parameters | Type | Description |
|------------------------|----------------------------|---|------|--|
| printClassReport | /classId/print/file | -filter String -sort String -limit Long -start Long -extension String -attributes String | GET | Obtain the result of a specific report with the provided extension |
| printClassSchemaReport | /classId/print_schema/file | -file String -extension String | GET | Obtain a file with the schema of the specified class |
| printSchemaReport | /print_schema/file | -file String -extension String | GET | Obtain a file with the schema of the whole database |

5.2.27. Class stats

<http://hostname:port/cmdbuild/services/rest/v3/classes/classId>

| Function | Path | Parameters | Type | Description |
|-----------|------------|-----------------------------------|------|------------------------------|
| stats | /stats | WsQueryParams -Query String | GET | Obtain class stats |
| relations | /relations | WsQueryOptions | GET | Obtain class relations stats |

5.2.28. Classes

http://hostname:port/cmdbuild/services/rest/v3/classes

Class data structure:

- name String
- description String
- defaultFilter Long
- defaultImportTemplate Long
- defaultExportTemplate Long
- _icon Long
- validationRule String
- type String
- messageAttr String
- flowStatusAttr String
- engine String
- parent String
- active Boolean
- prototype Boolean
- notInline Boolean
- notInlineClosed Boolean
- attachmentsInLine Boolean
- enableSaveButton Boolean
- attachmentTypeLookup String
- attachmentDescriptionMode String
- multitenantMode String
- stoppableByUser Boolean
- defaultOrder List<>
- formTriggers List<>
- contextMenuItems List<>
- widgets List<>
- attributeGroups List<>
- domainOrder List<String>
- formStructure JsonNode

| Function | Path | Parameters | Type | Description |
|----------|------|----------------------|------|--|
| readAll | | -detailed Boolean | GET | Obtain a list of every available class |

| | | | | |
|--------|----------|---|--------|---|
| | | -limit Long -start Long -filter String | | |
| read | /classId | | GET | Get the details of a specific class |
| create | | -data json | POST | Create a new class with the provided data |
| update | /classId | -data json | PUT | Update an existing class with the provided data |
| delete | /classId | | DELETE | Delete a specific class |

5.2.29. Configurations

<http://hostname:port/cmdbuild/services/rest/v3/configuration>

| Function | Path | Parameters | Type | Description |
|-----------------|---------|------------|------|---------------------------------|
| getPublicConfig | /public | | GET | Obtain the public configuration |
| getSystemConfig | /system | | GET | Obtain the system configuration |

5.2.30. Context menu component

<http://hostname:port/cmdbuild/services/rest/v3/components/contextmenu>

Custom menu component data structure:

- description String
- active boolean
- device TargetDevice

| Function | Path | Parameters | Type | Description |
|----------|----------|----------------------|--------|--|
| list | | | GET | Obtain a full list of the available custom menu components |
| get | /id | | GET | Obtain the details of a specific custom menu component |
| delete | /id | | DELETE | Delete a specific custom menu component |
| download | /id/file | -extension String | GET | Download the custom menu component |

| | | | | |
|--------|-----|--|------|--|
| | | -parameters String | | |
| create | | -file dataHandler -data json -merge Boolean | POST | Create a new custom menu component with the provided data |
| update | /id | -file dataHandler | PUT | Update a specific custom menu component with the provided file |
| update | /id | -data json | PUT | Update a specific custom menu component with the provided data |

5.2.31. Custom pages

http://hostname:port/cmdbuild/services/rest/v3/custompages

Custom page data structure:

- description String
- active boolean
- device TargetDevice

| Function | Path | Parameters | Type | Description |
|----------|----------|--|--------|--|
| list | | | GET | Obtain a full list of the available custom pages |
| get | /id | | GET | Obtain the details of a specific custom page |
| delete | /id | | DELETE | Delete a specific custom page |
| create | | -file dataHandler -data json -merge Boolean | POST | Create a new custom page with the provided data |
| update | /id | -file dataHandler | PUT | Update a specific custom page with the provided data |
| update | /id | -data json | PUT | Update a specific custom page with the provided data |
| download | /id/file | -extension String -parameters String | GET | Download the custom page file |

5.2.32. Dashboard

<http://hostname:port/cmdbuild/services/rest/v3/dashboards>

Dashboard data structure:

- name String
- description String
- active boolean
- charts Object
- layout Object

| Function | Path | Parameters | Type | Description |
|----------|------|--|--------|---|
| getAll | | -detailed Boolean -limit Integer -start Integer | GET | List all available dashboards |
| readOne | /id | | GET | Obtain a specific dashboard |
| create | | -data json | POST | Create a new dashboard with the provided data |
| update | /id | -data json | PUT | Update an existing dashboard with the provided data |
| delete | /id | | DELETE | Delete a specific dashboard |

5.2.33. Dms category values

<http://hostname:port/cmdbuild/services/rest/v3/dms/categories/lookupTypeId/values>

Same data structure as the LookUp values one.

| Function | Path | Parameters | Type | Description |
|----------|----------------|-------------------------------|--------|--|
| read | /lookupValueId | | GET | Obtain a specific dms category value |
| readAll | | | GET | Obtain a list of all the dms category values for the specific category |
| create | | -data json | POST | Create a new dms category value |
| update | /lookupValueId | -data json | PUT | Update an existing dms category value |
| delete | /lookupValueId | | DELETE | Delete an existing dms category value |
| reorder | /order | -lookupValueIds List<Long> | POST | Reorder the dms category values |

5.2.34. Dms categories

<http://hostname:port/cmdbuild/services/rest/v3/dms/categories>

Same data structure as the LookUp type one

| Function | Path | Parameters | Type | Description |
|------------------|----------------|---------------|--------|---|
| read | /lookupValueId | | GET | Obtain a specific dms category |
| readAll | | | GET | Obtain a list of all the dms categories |
| create | | -data json | POST | Create a new dms category |
| deleteLookupType | | | DELETE | Delete a dms category |

5.2.35. Dms models

<http://hostname:port/cmdbuild/services/rest/v3/dms/models>

Same data structure as the class one

| Function | Path | Parameters | Type | Description |
|------------------------|--------------------------------|----------------------------|--------|--|
| readAll | | WsQueryOptions | GET | Obtain all dms models |
| read | /classId | | GET | Obtain a specific dms model |
| create | | -data json | POST | Create a new dms model |
| update | /classId | -data json | PUT | Update an existing dms model |
| delete | /classId | | DELETE | Delete an existing dms model |
| readAttribute | /classId/attributes/ attrId | | GET | Read a specific attribute of the specified dms model |
| readAllAttributes | /classId/attributes | WsQueryOptions | GET | Read all attributes of a specific dms model |
| createAttribute | /classId/attributes | -data json | POST | Create a new attribute for a specific dms model |
| updateAttributes | /classId/attributes/ attrId | -data json | PUT | Update a specific attribute for a dms model |
| deleteAttributes | /classId/attributes/ attrId | | DELETE | Delete a specific attribute |
| reorderAttributes | /classId/attributes/ order | -attrOrder List<String> | POST | Reorder the dms model attributes |
| printModelSchemaReport | /classId/ print_schema/file | -extension String | GET | Print the schema of a specific dms model |

5.2.36. Domain attributes

<http://hostname:port/cmdbuild/services/rest/v3/domains/domainId/attributes>

Attribute data structure:

- formatPattern String
- unitOfMeasure String
- unitOfMeasureLocation String
- visibleDecimals Integer
- preselectIfUnique boolean
- showThousandsSeparator boolean
- showSeconds boolean
- showSeparators boolean
- type String
- name String
- description String
- showInGrid boolean
- showInReducedGrid boolean
- domainKey String
- domain String
- direction String
- unique boolean
- mandatory boolean
- active boolean
- index Integer
- defaultValue String
- group String
- precision String
- scale String
- targetClass String
- maxLenght Integer
- editorType String
- lookupType String
- filter String
- help String
- showIf String
- validationRules String

- mode boolean
- autoValue String
- metadata map<String, String>
- classOrder Integer
- isMasterDetail Boolean
- masterDetailDescription String
- ipType String
- textContentSecurity String

| Function | Path | Parameters | Type | Description |
|----------|---------|--|--------|--|
| readAll | | -limit Integer -start Integer | GET | Obtain the full attributes list of a specific domain, with the possibility of filtering the result with limit and offset |
| read | /attrId | | GET | Get the details of a specific domain attribute |
| create | | -data json | POST | Create a new attribute for a specific domain with the provided data |
| update | /attrId | -data json | PUT | Update an existing attribute with the provided data |
| delete | /attrId | | DELETE | Delete a specific attribute owned by a domain |
| reorder | /order | -attrOrder List <String> | POST | Reorder the list of domain attributes of a specific domain with the provided order provided in attrOrder |

5.2.37. Domains

http://hostname:port/cmdbuild/services/rest/v3/domains

Domain data structure:

- source String
- name String
- description String
- destination String
- cardinality String
- descriptionDirect String
- descriptionInverse String
- indexDirect int
- indexInverse int
- descriptionMasterDetail String

- filterMasterDetail String
- isMasterDetail boolean
- active boolean
- disabledSourceDescendants List<String>
- disabledDestinationDescendants List<String>
- inline Boolean
- defaultClosed Boolean

| Function | Path | Parameters | Type | Description |
|----------|-----------|---|--------|---|
| readAll | | -filter String -limit Integer -start Integer | GET | Obtain a complete list of all available domains with the possibility of filtering the results with limit and offset |
| read | /domainId | | GET | Get the details of a specific domain |
| create | | -data json | POST | Create a new domain with the provided data |
| update | /domainId | -data json | PUT | Update an existing domain with the provided data |
| delete | /domainId | | DELETE | Delete an existing domain |

5.2.38. Email accounts

http://hostname:port/cmdbuild/services/rest/v3/email/accounts

Email account data structure:

- name String
- username String
- password String
- address String
- smtp_server String
- smtp_port Integer
- smtp_ssl boolean
- smtp_starttls boolean
- imap_output_folder String
- imap_server String
- imap_port Integer
- imap_ssl boolean
- imap_starttls boolean

- bcc String
- subject String
- body String
- contentType String
- account String
- keepSynchronization boolean
- promptSynchronization boolean
- delay Long
- data Map<String, String>

| Function | Path | Parameters | Type | Description |
|----------|-------------|---|--------|--|
| readAll | | -filter String -sort String -limit Long -start Long -detailed Boolean -includeBindings Boolean | GET | Obtain a full list of available email templates with the possibility of filtering the results with limit offset and sort |
| read | /templateId | | GET | Get the details of a specific template |
| create | | -data json | POST | Create a new email template with the provided data |
| update | /templateId | -data json | PUT | Update an existing template with the provided data |
| delete | /templateId | | DELETE | Delete an existing email template |

5.2.41. Etl Gate

<http://hostname:port/cmdbuild/services/rest/v3/etl/gates>

Etl gate data structure:

- code String
- processingMode String
- allowPublicAccess Boolean
- enabled Boolean
- config Map<String, String>
- template String
- script String

| Function | Path | Parameters | Type | Description |
|-----------------|------------------|---|--------|--|
| readAll | | -limit Long -offset Long -detailed boolean | GET | Obtain a list of all Etl gates |
| readAllForClass | by-class/classId | -wsQueryOptions - include_etl_templates boolean | GET | Obtain a list of etl gates for a class |
| read | /gateId | | GET | Get the details of a specific etl gate |
| create | | -date json | POST | Create a new Etl gate with the provided data |
| update | /gateId | -data json | PUT | Update an existing Etl gate with the provided data |
| delete | /gateId | | DELETE | Remove an existing Etl gate |

5.2.42. Etl templates

<http://hostname:port/cmdbuild/services/rest/v3/etl/templates>

Etl template data structure:

- errorMessageTemplate Long
- notificationEmailTemplate Long
- errorMessageAccount Long
- fileFormat String
- code String
- description String
- targetName String
- targetType String
- source String
- exportFilter String
- mergeMode String
- mergeMode_when_missing_update_attr String
- mergeMode_when_missing_update_value String
- active Boolean
- type String
- useHeader Boolean
- ignoreColumnOrder Boolean

- headerRow Integer
- dataRow Integer
- firstCol Integer
- charset String
- csv_separator String
- importKeyAttribute String
- importKeyAttributes List<String>
- filter JsonNode
- columns List<EtlCoulmnData>
- dateFormat String
- timeFormat String
- decimalSeparator String
- dateTimeFormat String
- thousandsSeparator String
- attributes List<Attributes>

| Function | Path | Parameters | Type | Description |
|-----------------------------|---|--|--------|---|
| readAll | | -WsQueryOptions | GET | Obtain all etl templates |
| readAllForClass | /by-class/classId | -WsQueryOptions -include_related_domains boolean | GET | Obtain all etl templates for a specific class |
| readOne | /templateId | | GET | Get the details of a specific template |
| executeExportTemplate | /templateId/export /templateId/export/ fileName | -filterStr String | GET | Execute an export using a template |
| executeImportTemplate | /templateId | -dile dataHandler -detailed_report boolean | POST | Execute an import using a template |
| create | | -data json | POST | Create a new etl template |
| update | /templateId | -data json | PUT | Update an existing etl template |
| delete | /templateId | | DELETE | Delete an existing template |
| executeInlineExportTemplate | /inline/export /inline/export/ fileName | -data String -config json | POST | |

| | | | | |
|-----------------------------|----------------|---|------|--|
| executeInlineImportTemplate | /inline/import | -file dataHandler -config json | POST | |
|-----------------------------|----------------|---|------|--|

5.2.43. Fk Domain

<http://hostname:port/cmdbuild/services/rest/v3/fkdomains>

| Function | Path | Parameters | Type | Description |
|----------|------|---|------|--|
| readAll | | -filter String -limit Long -start Long | GET | Obtain a list of all FKs for every domain, with the possibility of filtering the results |

5.2.44. Functions

<http://hostname:port/cmdbuild/services/rest/v3/functions>

| Function | Path | Parameters | Type | Description |
|----------------------|------------------------|---|------|---|
| readAll | | -limit Integer -start integer -filter String -detailed Boolean | GET | Obtain a list of all available functions with the possibility of filtering the results with limit offset and filter |
| read | /functionId | | GET | Get the details of a specific functions |
| readInputParameters | /functionId/parameters | -limit Integer -start Integer | GET | Get a list of input parameters of a specific function |
| readOutputParameters | /functionId/attributes | -limit Integer -start Integer | GET | Get a list of output parameters of a specific function |
| call | /functionId/outputs | -parameters String -model String | GET | Call a specific function |

5.2.45. Geo attributes

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/geoattributes>

Geo attribute data structure:

- name String
- subtype String
- description String
- index int
- visibility List
- zoomMin int
- zoomMax int
- zoomDef int
- style Map<String, Object>
- _icon Long

| Function | Path | Parameters | Type | Description |
|-------------------|--------------|---|--------|---|
| readAllAttributes | | -limit Integer -start Integer -detailed Boolean -visible Boolean | GET | Obtain a full list of all available attributes with the possibility of filtering the results with offset, limit and visible |
| reorder | /order | -attrOrder List<Long> | POST | Reorder the list of attributes with the details provided in attrOrder |
| readAttribute | /attributeId | | GET | Get the details of a specific attribute |
| create | | -data json | POST | Create a new attribute for a specific class with the provided data |
| updateVisibility | /visibility | -geoAttributes List<Long> | POST | Update the visibility of the attributes owned by a specific class |
| update | /attributeId | -data json | PUT | Update a specific attribute with the provided data |
| delete | /attributeId | | DELETE | Delete a specific attribute |

5.2.46. Geo style rules

<http://hostname:port/cmdbuild/services/rest/v3/processes|classes/classId/geostylerrules>

Geo style rules data structure:

- name String
- description String
- owner String
- attribute String
- classattribute String
- function String
- alanalysistype String
- segments Integer
- rules JsonNode

| Function | Path | Parameters | Type | Description |
|------------|-------------------|--|--------|---|
| readAll | | -limit Integer -start Integer | GET | Obtain a full list of geo style rules |
| read | /rulesetId | | GET | Get the details of a specific geo style rule owned by the class with id classId |
| create | | -data json | POST | Create a new geo style ruleset with the provided data for the class with id classId |
| update | /rulesetId | -data json | PUT | Update a specific geo style rule with the provided data |
| delete | /rulesetId | | DELETE | Delete an existing ruleset with the id rulesetId and class with id classId |
| applyRules | /rulesetId/result | -cards String | GET | Get the results of the application of a ruleset with id rulesetId |
| applyRules | /tryRules | -data json -cards String | POST | Get the results of the application of a ruleset with the provided data |

| | | | | |
|---------------|----------|---------------------------------------|--------|---|
| getAllForCard | | -visible Boolean | GET | Obtain a full list of geo server layers for a specific card |
| create | | -data json -file DataHandler | POST | Create a new attribute with the provided in data for a specific card |
| update | /layerId | -data json -file DataHandler | PUT | Update an existing geo server layer with the provided data |
| delete | /layerId | | DELETE | Delete a specific geo server layer |
| order | /order | -layerOrder List<Long> | POST | Reorder the list of layers for a specific class based on the details provided in layerOrder |

5.2.49. Grants

<http://hostname:port/cmdbuild/services/rest/v3/roles/roleId/grants>

Grant data structure:

- mode String
- objectType String
- objectTypeName String
- filter String
- attributePrivileges Map<String, String>
- _card_create_disabled Boolean
- _card_update_disabled Boolean
- _card_delete_disabled Boolean
- _card_clone_disabled Boolean
- _card_relation_disabled Boolean
- _card_print_disabled Boolean
- _can_fc_attachment Boolean
- _on_filter_mismatch Boolean

| Function | Path | Parameters | Type | Description |
|----------|------|---|------|---|
| readMany | | -filter String -limit Long -start Long | GET | Obtain a list of available grants for a specific role |

| | | | | |
|-----------------|-------------------|----------------------------------|--------|--|
| | | json | | provided data |
| update | /jobId | -data json | PUT | Update an existing job with the provided data |
| delete | /jobId | | DELETE | Delete a specific existing job |
| runJobNow | /jobId/run | | POST | Run a specific job |
| getJobRuns | /jobId/runs | -limit Long -start Long | GET | Get a list of every execution of a specific job |
| getJonRunErrors | /jobId/errors | -limit Long -start Long | GET | Get a list of all the errors generated by a specific job |
| getJobRuns | /_ANY/runs | -limit Long -start Long | GET | Get a list of every run of every job |
| getJonRunErrors | /_ANY/errors | -limit Long -start Long | GET | Get a list of all the errors generated by all jobs |
| getJobRuns | /jobId/runs/runId | | GET | Get the details of a specific run of a specific job |

5.2.52. Language configurations

<http://hostname:port/cmdbuild/services/rest/v3/configuration/languages>

| Function | Path | Parameters | Type | Description |
|-------------------|------|------------|------|--|
| getLoginLanguages | | | GET | Obtain a list of all available languages |

5.2.53. Languages

<http://hostname:port/cmdbuild/services/rest/v3/languages>

| Function | Path | Parameters | Type | Description |
|---------------|------|--------------------|------|--|
| readLanguages | | -active Boolean | GET | Obtain a list of all available languages |

5.2.54. Locks

<http://hostname:port/cmdbuild/services/rest/v3/locks>

| Function | Path | Parameters | Type | Description |
|----------------|---------|------------|--------|--------------------------------|
| getLocks | | | GET | Get all available locks |
| getLock | /lockId | | GET | Get details of a specific lock |
| deleteLock | /lockId | | DELETE | Delete a specific lock |
| deleteAllLocks | /_ANY | | DELETE | Delete all locks |

5.2.55. Lookup types

http://hostname:port/cmdbuild/services/rest/v3/lookup_types

Lookup type data structure:

- name String
- parent String

| Function | Path | Parameters | Type | Description |
|------------------|---------------|---|--------|---|
| read | /lookupTypeId | | GET | Get the details of a specific lookup type |
| readAll | | -limit Long -start Long -filter String | GET | Obtain a list of all available lookup types |
| createLookupType | | -data json | POST | Create a new lookup type with the provided data |
| updateLookupType | /lookupTypeId | -data json | PUT | Update an existing lookup type with the values provided in wsLookupType |
| deleteLookupType | /lookupTypeId | | DELETE | Delete an existing lookup type |

5.2.56. Lookup values

http://hostname:port/cmdbuild/services/rest/v3/lookup_types/lookupTypeId/values

Lookup data structure:

- code String
- description String
- index Integer
- active boolean

- parent_id Long
- default boolean
- note String
- text_color String
- icon_type String
- icon_image String
- icon_font String
- icon_color String

| Function | Path | Parameters | Type | Description |
|----------|----------------|---|--------|--|
| read | /lookupValueId | | GET | Obtain a list of the lookup values for a specific lookup type |
| readAll | | -limit Long -start Long -filter String -active Boolean | GET | Obtain a full list of the lookup values available |
| create | | -data json | POST | Create a new lookup value for a specific lookup type with the provided data |
| update | /lookupValueId | -data json | PUT | Update an existing lookup value with the provided data |
| delete | /lookupValueId | | DELETE | Delete an existing lookup value |
| reorder | /order | -lookupValueIds List<Long> | POST | Reorder the lookup value list of a specific lookup type with the provided data |

5.2.57. Menu

<http://hostname:port/cmdbuild/services/rest/v3/menu>

Menu Node data structure:

- _id String
- menuType MenuItemType
- objectTypeName String
- objectDescription String
- children List<MenuNodes>

- direction String
- recursionEnabled Boolean
- showOnlyOne Boolean
- nodes List<TreeNodees>

| Function | Path | Parameters | Type | Description |
|----------|---------|---|--------|--|
| readAll | | -filter String -limit Long -start Long | GET | Get a list of every available nav tree |
| read | /treeId | -treeMode String | GET | Obtain the details of a specific nav tree |
| create | | -data json | POST | Create a new nav tree with the provided data |
| update | /treeId | -data json | PUT | Update an existing nav tree with the provided data |
| delete | /treeId | | DELETE | Delete a specific nav tree |

5.2.60. Process configuration

<http://hostname:port/cmdbuild/services/rest/v3/configuration/processes>

| Function | Path | Parameters | Type | Description |
|--------------|-----------|------------|------|--|
| readStatuses | /statuses | | GET | Obtain a list with the statuses of every available process |

5.2.61. Process instance activity email

<http://hostname:port/cmdbuild/services/rest/v3/processes/processId/instances/instanceId/activities/activityId/emails>

| Function | Path | Parameters | Type | Description |
|-------------------------|-------|------------|------|---|
| updateEmailWithCardData | /sync | -flowData | POST | Update a specific email with a specific card data |

5.2.62. Process instance activity

<http://hostname:port/cmdbuild/services/rest/v3/processes/processId/instances/processInstanceId/activities>

| Function | Path | Parameters | Type | Description |
|----------|------|------------|------|--|
| read | | | GET | Obtain a list of every available task for a specific |

| | | | | |
|------|--------------------|--|-----|--|
| | | | | class |
| read | /processActivityId | | GET | Get all the details of a specific task |

5.2.63. Process instance history

http://hostname:port/cmdbuild/services/rest/v3/processes/processId/instances/instanceId/history

| Function | Path | Parameters | Type | Description |
|------------------|-----------|----------------------------------|------|---|
| getHistory | | -limit Long -start Long | GET | Obtain a list of the history of processes for a specific card |
| getHistoryRecord | /recordId | | GET | Get the details of a specific record in the history |

5.2.64. Process instances

http://hostname:port/cmdbuild/services/rest/v3/processes/processId/instances

| Function | Path | Parameters | Type | Description |
|-----------|------------------------------|--|--------|---|
| create | | -data json | POST | Create a new process instance with the provided data |
| update | /processInstanceId | -data json | PUT | Update a process instance with the provided data |
| read | /processInstanceId | -include_tasklist Boolean | GET | Get the details of a specific process |
| plotGraph | /processInstanceId /graph | -simplified Boolean | GET | |
| readMany | | -filter String -sort String -limit Long -start Long -positionOf Long -positionOf_goToPage Boolean -include_tasklist Boolean | GET | Obtain the list of all available process instances with the possibility of filtering the results with filter, sort, limit, offset, positionOfCard, goToPage |
| delete | /processInstanceId | | DELETE | Delete an existing process instance |

5.2.65. Process start activities

http://hostname:port/cmdbuild/services/rest/v3/processes/processId/start_activities

| Function | Path | Parameters | Type | Description |
|----------|------|------------|------|--|
| read | | | GET | Get the start activities of a specific process |

5.2.66. Process task definition

http://hostname:port/cmdbuild/services/rest/v3/processes/processId/activities

Task definition data structure:

- formStructure JsonNode

| Function | Path | Parameters | Type | Description |
|------------------|---------|----------------------------------|------|---|
| getAllActivities | | -limit Long -start Long | GET | Get all task definitions for a specific process |
| getOne | /taskId | | GET | Obtain the details of a specific task |
| update | /taskId | -data json | PUT | Update an existing task |

5.2.67. Process task

http://hostname:port/cmdbuild/services/rest/v3/processes/processId/instance_activities

| Function | Path | Parameters | Type | Description |
|------------------|------|----------------------------------|------|--|
| getAllActivities | | -limit Long -start Long | GET | Get all activities of a specific process |

5.2.68. Processes

http://hostname:port/cmdbuild/services/rest/v3/processes

| Function | Path | Parameters | Type | Description |
|----------|------------|--|------|---|
| readAll | | -activeOnly Boolean -limit Long -start Long -detailed Boolean | GET | Obtain a list of every available process |
| read | /processId | | GET | Get the details of a specific process |
| create | | -data json | POST | Create a new process with the provided data |

| | | | | |
|--|-------------------------------------|---|--------|---|
| update | /processId | -data json | PUT | Update an existing process with the provided data |
| delete | /processId | | DELETE | Delete an existing process |
| uploadNewXpdVersion | /processId/ versions | -file DataHandler | POST | Upload a new xpd |
| uploadXpdVersionAndMigrateProcessToNewProvider | /processId/ migration | -provider String -file DataHandler | POST | Upload a new xpd and force the process to use the new xpd version |
| getAllXpdVersions | /processId/ versions | | GET | Obtain a list of all xpd version |
| getXpdVersionFile | /processId/ versions/planId/file | | GET | Obtain an xpd file |
| getXpdTemplateFile | /processId/ template | | GET | Obtain an xpd template file |

5.2.69. Relations

http://hostname:port/cmdbuild/services/rest/v3/domains/domainId/relations

Relation data structure:

- `_id` Long
- `_type` String
- `_sourceType` String
- `_sourceId` Long
- `_destinationType` String
- `_destinationId` Long
- `_is_direct` Boolean

| Function | Path | Parameters | Type | Description |
|-------------------|-------------|--|--------|--|
| readAll | | -limit Long -start Long -detailed Boolean | GET | Obtain a list of all available relations for a specific domain |
| read | /relationId | | GET | Get the details of a specific relation |
| create | | -data json | POST | Create a new relation with the provided data |
| update | /relationId | -data json | PUT | Update an existing relation with the provided data |
| delete | /relationId | | DELETE | Delete a specific relation |
| moveManyRelations | /_ANY/move | -data json | POST | Move the specified relation |

| | | | | |
|-------------------|------------|------------|------|--------------------------|
| copyManyRelations | /_ANY/copy | -data json | POST | Copy a specific relation |
|-------------------|------------|------------|------|--------------------------|

5.2.70. Reports

http://hostname:port/cmdbuild/services/rest/v3/reports

Report data structure:

- `_id` Long
- `code` String
- `description` String
- `_description_translation` String
- `active` boolean
- `title` String
- `query` String

| Function | Path | Parameters | Type | Description |
|----------------------|---|---|------|--|
| readAll | | -filter String -limit Long -start Long -detailed Boolean | GET | Obtain a list of all available reports |
| read | /reportId | | GET | Get the details of a specific report |
| readAllAttributes | /reportId/attributes | -limit Long -start Long | GET | Get all attributes of a specific report |
| download | /reportId/file: | -extension String -parameters String | GET | Download a report on a specified file with a specified extension |
| createReport | | -data json -attachments List<Attachment> | POST | Create a new report with the provided attachments |
| updateReport | /reportId | -attachments List<Attachment> | PUT | Update an existing report with the provided data |
| updateReportTemplate | /reportId/template-data json -attachments List<Attachment> | -data json -attachments List<Attachment> | PUT | Update an existing report template with the provided data |

| | | | | |
|-----------------|---------------|--|------|--|
| readRoleUser | /roleId/users | -filter String -sort String -limit Long -start Long -assigned Boolean | GET | Obtain a list of all available roles for a specific user |
| updateUsersPost | /roleId/users | -users json | POST | Update the roles of a specific user |
| create | | -jsonData String | POST | Create a new role with the provided data |
| update | /roleId | -jsonData String | PUT | Update an existing role with the provided data |

5.2.74. Session menu

<http://hostname:port/cmdbuild/services/rest/v3/sessions/sessionId/menu>

| Function | Path | Parameters | Type | Description |
|----------|------|------------------|------|---|
| read | | -flat Boolean | GET | Obtain a list of every session menu available |

5.2.75. Session preferences

<http://hostname:port/cmdbuild/services/rest/v3/sessions/sessionId/preferences>

| Function | Path | Parameters | Type | Description |
|-------------------------|------|------------------------------------|--------|---|
| read | | | GET | Obtain a list of all available preferences for a specific session |
| getUserConfig | /key | -key String | GET | Get the value of a specific user configuration |
| updateUserConfigValue | /key | -key String -value String | PUT | Update the value of a specific user configuration |
| updateUserConfigValues | | -data Map<String, String> | POST | Update the vlue of multiple user configurations |
| deleteSystemConfigValue | /key | | DELETE | Delete a specific configuration key |

5.2.76. Sessions

<http://hostname:port/cmdbuild/services/rest/v3/sessions>

Session data structure:

- username String
- password String
- role String
- scope String
- tenant Long
- ignoreTenants Boolean
- activeTenants List<Long>

| Function | Path | Parameters | Type | Description |
|-----------|---------------------------|---|--------|---|
| create | | -data json -includeExtendedData Boolean -scopeStr String -returnId Boolean | POST | Create a new session with the provided data |
| read | /sessionId | -include ExtendedData Boolean | GET | Get the details of a specific session |
| read | /sessionId/ privileges | | GET | Get the details of the privileges of a specific session |
| readAll | | | GET | Obtain a list of all available sessions |
| update | /sessionId | -data json -includeExtendedData Boolean | PUT | Update an existing session with the provided data |
| delete | /sessionId | | DELETE | Delete an existing session |
| deleteAll | /all | | DELETE | Delete all existing sessions |
| keepAlive | /id/keepalive | | POST | Keep a session alive |

5.2.77. System configuration

<http://hostname:port/cmdbuild/services/rest/v3/system/config>

| Function | Path | Parameters | Type | Description |
|-----------------|------|----------------------|------|-----------------------------------|
| getSystemConfig | | -detailed Boolean | GET | Get the full system configuration |

| | | | | |
|--------------------------|---------|---|--------|--|
| getSystemConfig | /key | -includeDefault Boolean | GET | Get a specific configuration of the system |
| updateSystemConfigValue | /key | -value String -encrypt Boolean | PUT | Update a specific configuration of the system |
| updateSystemConfigValues | /_MANY | -data Map<String, String> | PUT | Update the system configuration with the provided data |
| deleteSystemConfigValue | /key | | DELETE | Delete a specific system configuration |
| reloadConfig | /reload | | POST | Reload the system configuration |

5.2.78. System

http://hostname:port/cmdbuild/services/rest/v3/system

| Function | Path | Parameters | Type | Description |
|------------------|-------------------------|----------------------|------|--|
| status | /status | | GET | Obtain a list of the system services status |
| getClusterStatus | /cluster/status | | GET | Get the status of the cluster system |
| dropCache | /cache/drop | | POST | Invalidate all system cache |
| dropCache | /cache/ cacheId/drop | | POST | Invalidate a specific cache |
| getCacheStats | /cache/stats | | GET | Obtain a list of the cache statuses |
| stopSystem | /stop | | POST | Stop the CMDBuild system |
| reloadSystem | /reload | | POST | Reload the CMDBuild system |
| restartSystem | /restart | | POST | Restart the CMDBuild system |
| upgradeSystem | /upgrade | -file DataHandler | POST | Upgrade the CMDBuild system with the provided file |
| dropAudit | /audit/drop | | POST | Drop the audits of the system |
| cleanupAudit | /audit/cleanup | | POST | Cleanup the audits of the system |
| getAllPatches | /patches | | GET | Obtain a list of all patches |
| getAllTenants | /tenants | | GET | Obtain a list of all available tenants |
| getSchedulerJobs | /scheduler/ jobs | | GET | Obtain a list of all the jobs available in the scheduler |
| triggerJobNow | /scheduler/ | | POST | Trigger the execution of a |

| | | | | |
|--------------------------|---------------------------|---|--------|--|
| | job/jobId/ trigger | | | job |
| getAllLoggers | /loggers | | GET | Obtain a list of all available loggers |
| updateLoggerLevel | /loggers/key | -loggerCategory String -loggerLevel String | POST | Update the level of a specific logger |
| addLoggerLevel | /loggers/key | -loggerCategory String -loggerLevel String | PUT | Add a logger level to an existing logger |
| deleteLoggerLevel | /loggers/key | -loggerCategory String | DELETE | Delete a level of a specific logger |
| receiveLogMessages | /logger/stream | | POST | Enable the stream of the active loggers |
| stopReceivingLogMessages | /logger/stream | | DELETE | Stop the stream of the active loggers |
| dumpDatabase | /database/ dump | | GET | Create a dump of the current database used by the system |
| reconfigureDatabase | /database/ reconfigure | -dbConfig Map<String, String> | POST | Reconfigure the database with the configuration provided by dbConfig |
| importDatabaseFromDump | /database/ import | -file DataHandler | POST | Reconfigure a database with the dump file provided by dataHandler |
| generateDebugInfo | /debuginfo/ download | | GET | Generate a bug report |
| sendBugReport | /debuginfo/ send | -message String | POST | Send a bug report with the provided message |
| sendBroadcastMessage | /messages/ broadcast | -message String | POST | Send a broadcast alert with the provided message |

5.2.79. Tenants

<http://hostname:port/cmdbuild/services/rest/v3/tenants>

| Function | Path | Parameters | Type | Description |
|----------------------|------------|----------------------------------|------|--|
| getAll | | -limit Long -start Long | GET | Obtain a list of all available tenants |
| configureMultitenant | /configure | -configData json | POST | Set the multitenant configuration with the provided data |

5.2.80. Timezones

http://hostname:port/cmdbuild/services/rest/v3/timezones

| Function | Path | Parameters | Type | Description |
|------------------------|------|------------|------|--|
| readAvailableTimezones | | | GET | Obtain a list of all available timezones |

5.2.81. Translations

http://hostname:port/cmdbuild/services/rest/v3/translations

| Function | Path | Parameters | Type | Description |
|-----------------------------|----------|---|--------|---|
| getAll | | -limit Long -start Long -filter String | GET | Obtain a list of all available translations |
| getAllAggregateByCode | /by-code | WsQueryOptions -lang String -includeRecordsWithoutTranslation Boolean -section String | GET | Obtain a list of specific translations, divided by language and section |
| getTranslationForKeyAndLang | /code | -lang String | GET | Get the translation for a specific key in a specific language |
| setTranslation | /code | -data json | PUT | Set a specific translation with the provided data |
| deleteTranslation | /code | -lang String | DELETE | Remove a specific translation for a specific language |
| export | /export | -language String -format String -filter String -separator String | GET | Obtain a file of the specified format containing the translations for a specific language |
| import | | -separator String -file dataHandler | POST | Upload a file containing translations |

5.2.82. Uploads

<http://hostname:port/cmdbuild/services/rest/v3/uploads>

Upload item data structure:

- path String
- description String

| Function | Path | Parameters | Type | Description |
|-------------------|------------------|--|--------|--|
| readMany | | -dir String | GET | Obtain a list of all available files in a specific directory |
| readFile | /fileId | | GET | Get the details of a specific file |
| downloadFile | /fileId/file: | | GET | Download a specific file |
| downloadManyFiles | /_MANY/file:.zip | -dir | GET | Download multiple specified files |
| downloadAllFiles | /_ANY/file:.zip | | GET | Download all available files |
| loadFile | | -file DataHandler -directoryPath String | POST | Load a new file in the system |
| loadZipFile | /_MANY | -dataHandler DataHandler | POST | Load a zip file containing one or more files |
| updateFile | /fileId | -file DataHandler | PUT | Update an existing file with the provided data |
| deleteFile | /fileId | | DELETE | Delete a specific file |

5.2.83. Users

<http://hostname:port/cmdbuild/services/rest/v3/users>

User data structure:

- username String
- description String
- email String
- password String
- initialPage String
- active boolean
- service boolean
- language String
- multiGroup boolean
- multiTenant boolean

- multiTenantActivationPrivileges String
- defaultUserGroup Long
- userTenants List<TenantInfo>
- userGroups List<UserRole>

| Function | Path | Parameters | Type | Description |
|----------------|-----------------------|--|------|--|
| readMany | | -filter String -sort String -limit Long -start Long -detailed Boolean | GET | Obtain a list of all available users with the possibility of filtering the results with filter, sort, limit and offset |
| readOne | /userId | | GET | Get the details of a specific user |
| create | | -data json | POST | Create a new user with the provided data |
| update | /userId | -data json | PUT | Update an existing user with the provided data |
| changePassword | /current/ password | -data json | PUT | Update the password of the current user with the provided data |

5.2.84. Card views

<http://hostname:port/cmdbuild/services/rest/v3/views/viewId/cards>

| Function | Path | Parameters | Type | Description |
|----------|---------|---|------|--|
| create | | -data Map<String, Object> | POST | Create a new view with the provided data |
| readOne | /cardId | | GET | Get the card details with the specified view |
| readMany | | -filter String -sort String -limit Long -start Long -positionOf Long -forDomainName | GET | Obtain a list of information with the specified view |

5.3. REST Examples

In this paragraph various examples of REST calls will be presented. Note that CMDBuild in this scenario is configured with the database: demo.dump.xz, so if you want to replicate the same examples with the same data you must load that dump first.

To perform the following examples various tools can be used, via terminal with curl on linux operating systems, or with the support of a graphical interface with programs like Postman, or any other software that can perform HTTP requests.

Every request requires the user to specify in the header the field "Cmdbuild-authorization", that field is a session token generated when creating a session, the first example request will show how to obtain that through a specific request.

5.3.1. Generating a session token

The endpoint to use for generating a session token is:

`http://hostname:port/cmdbuild/services/rest/v3/sessions`

The request type has to be POST, because username and password will be provided to create a new session. If in the response we want to obtain the session token, from version 3.2 a query parameter has to be set to true in the request, the parameter is 'returnId'. If this parameter is not set to true the sessionId will be hidden and the value 'current' will be returned instead

The request will be like the following:

```
POST http://hostname:port/cmdbuild/services/rest/v3/sessions?
scope=service&returnId=true HTTP/1.1
Content-Type:application/json
{
  username : admin,
  password : admin
}
```

The response will be like the following, where the `_id` will be the generated session id and after the list of available roles information like multigroup and role privileges will be displayed:

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "success": true,
  "data" : {
    "_id": "sessionId",
    "username": "admin",
    "userDescription": "Administrator",
```

```
    "role": "SuperUser",
    "availableRoles": [
      "SuperUser"
    ],
    ...
  }
}
```

An inactive session will be deleted after a certain amount of time, causing the user to re-create the session every once in a while.

With further requests users can provide the generated id (the value in the field `_id`) in the header to obtain access to every rest endpoint.

5.3.2. Obtaining a list of every class

If the user wants to obtain a list of the available classes the endpoint that will be used is:

`http://hostname:port/cmdbuild/services/rest/v3/classes`

The request type has to be GET, and will look like the following:

```
GET http://hostname:port/cmdbuild/services/rest/v3/classes?scope=service
HTTP/1.1
Cmdbuild-authorization:sessionId
```

The response will be like the following, the results should be 24, but for the documentation purpose only the first results are displayed.

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "success": true,
  "data" : {
    "_id": "Invoice",
    "name": "Invoice",
    "description": "Invoice",
    "_description_translation": "Invoice",
    "prototype": false,
    "parent": "Class",
    "active": true,
    "type": "standard",
```

```
"_can_read": true,
"_can_create": true,
"_can_update": true,
"_can_clone": true,
"_can_delete": true,
"_can_modify": true,
"defaultFilter": null,
"description_attribute_name": "Description",
"metadata": {},
"_icon": null
},
. . .
,
"meta": {
  "total": 24
}
}
```

In the response of multiple items at the bottom of the response a “meta” field will always be provided, various information such as the number of total results can be found here.

Note that the parameters previously described in the documentation can be provided (in this case the available parameters are activeOnly, detailed, limit and offset). If, for example, we wanted the amount of results to be limited to two the request would look the same with the addition of the parameter in the endpoint like:

```
http://hostname:port/cmdbuild/services/rest/v3/classes?
scope=service&limit=2
```

The same with the addition of other parameters.

5.3.3. Obtaining the information of a specific class

If instead of a class list, the user wants to obtain the information of a specific class only, the endpoint will be the same as the full list with the addition of the classId in the path:

```
http://hostname:port/cmdbuild/services/rest/v3/classes/classId
```

Where the value of classId will be the value of the class that we want to obtain, for the example the class we use will be the one previously returned in the response (the class with _id=invoice), so the new request endpoint will be:

```
http://hostname:port/cmdbuild/services/rest/v3/classes/Invoice
```

The request type has to be GET, and will look like the following:

```
GET http://hostname:port/cmdbuild/services/rest/v3/classes/Invoice?
scope=service HTTP/1.1
Cmdbuild-authorization:sessionId
```

The response will contain the information of only that class like shown in the response with every class:

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "success": true,
  "data" : {
    "_id": "Invoice",
    "name": "Invoice",
    "description": "Invoice",
    "_description_translation": "Invoice",
    "prototype": false,
    "parent": "Class",
    "active": true,
    "type": "standard",
    . . .
  }
}
```

5.3.4. Creating a new class

If the objective of the request has to be the creation of a new class, the endpoint is:

`http://hostname:port/cmdbuild/services/rest/v3/classes`

The new class information have to be provided, in the header it is also required to add the content-type, as in the session creation and the request type will be POST:

```
POST http://hostname:port/cmdbuild/services/rest/v3/classes HTTP/1.1
Cmdbuild-authorization:sessionId
Content-Type:application/json
{
  "name": "testClass",
  "type": "standard"
}
```

In this case the class created has only the two basic information, the name and the type, in the request we can add whatever information we want that is supported by the class.

When the request is made the response will contain the newly added class:

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "success": true,
  "data": {
    "_id": "testClass",
    "name": "testClass",
    "description": "",
    "_description_translation": "",
    "prototype": false,
    "parent": "Class",
    "active": true,
    "type": "standard",
    . . .
```

So that if we would perform a get request for that specific class:

```
GET http://hostname:port/cmdbuild/services/rest/v3/classes/testClass?
scope=service HTTP/1.1
Cmdbuild-authorization:sessionId
```

We would obtain those information that just got added.

5.3.5. Update an existing class

If a class has been already created, there is the possibility of updating the information of this class via a PUT request, the endpoint will be

`http://hostname:port/cmdbuild/services/rest/v3/classes/classId`

And in the request every element that needs changing can be included, for example if the objective is change the description of our previously created class (testClass) to "test description" this will be the request:

```
PUT http://hostname:port/cmdbuild/services/rest/v3/classes/testClass
HTTP/1.1
Cmdbuild-authorization:sessionId
Content-Type:application/json
{
  "name": "testClass",
```

```
    "type": "standard",
    "description": "test description"
  }
```

The request will contain the information about the updated class:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "success": true,
  "data": {
    "_id": "testClass",
    "name": "testClass",
    "description": "test description",
    "_description_translation": "",
    "prototype": false,
    "parent": "Class",
    "active": true,
    "type": "standard",
    . . .
  }
}
```

So that if we would perform a get request for that specific class:

```
GET http://hostname:port/cmdbuild/services/rest/v3/classes/testClass?
scope=service HTTP/1.1
Cmdbuild-authorization:sessionId
```

We would obtain those information that just got added.

6. Appendix: Glossary

6.1.1. ATTACHMENT

An attachment is a file associated to a card.

In order to manage the attachments, CMDBuild uses in embedded mode any document system which is compatible with the standard protocol CMIS (or the DMS Alfresco until the version 3 through its native webservice).

The management of the attachments supports the versioning of those files that have been uploaded a few times, with automatic numbering.

6.1.2. WORKFLOW STEP

"Activity" means one of the steps of which the process consists.

An activity has a name, an executor, a type, possible attributes and methods with statements (CMDBuild API) to be executed.

A process instance is a single process that has been activated automatically by the application or manually by an operator.

See also: Process

6.1.3. ATTRIBUTE

The term refers to an attribute of a CMDBuild class.

CMDBuild allows you to create new attributes (in classes and domains) or edit existing ones.

For example, in "supplier" class the attributes are: name, address, phone number, etc..

Each attribute corresponds, in the Management Module, to a form field and to a column in the database.

See also: Class, Domain, Report, Superclass, Attribute Type

6.1.4. BIM

Method with the aim to support the whole life cycle of a building: from its construction, use and maintenance, to its demolition, if any.

The BIM method (Building Information Modeling) is supported by several IT programs that can interact through an open format for data exchange, called IFC (Industry Foundation Classes).

See also: GIS

6.1.5. CI

We define CI (Configuration Item) each item that provides IT service to the user and has a sufficient detail level for its technical management.

CI examples include: server, workstation, software, operating system, printer, etc.

See also: Configuration

6.1.6. CLASS

A Class is a complex data type having a set of attributes that describe that kind of data.

A Class models an object that has to be managed in the CMDB, such as a computer, a software, a service provider, etc.

CMDBuild allows the administrator - with the Administration Module - to define new classes or delete / edit existing ones.

Classes are represented by cards and, in the database, by tables automatically created at the definition time.

See also: Card, Attribute

6.1.7. CONFIGURATION

The configuration management process is designed to keep updated and available to other processes the items (CI) information, their relations and their history.

It is one of the major ITIL processes managed by the application.

See also: CI, ITIL

6.1.8. DASHBOARD

In CMDBuild, a dashboard corresponds to a collection of different charts, in this way you can immediately hold in evidence some key parameters (KPI) related to a particular management aspect of the IT service.

See also: Report

6.1.9. DATABASE

The term refers to a structured collection of information, hosted on a server, as well as utility software that handle this information for tasks such as initialization, allocation, optimization, backup, etc..

CMDBuild relies on PostgreSQL, the most powerful, reliable, professional and open source database , and uses its advanced features and object-oriented structure.

6.1.10. DOMAIN

A domain is a relation between two classes.

A domain has a name, two descriptions (direct and inverse), classes codes, cardinality and attributes.

The system administrator, using the Administration Module, is able to define new domains or delete / edit existing ones.

It is possible to define custom attributes for each domain.

See also: Class, Relation

6.1.11. DATA FILTER

A data filter is a restriction of the list of those elements contained in a class, obtained by specifying boolean conditions (equal, not equal, contains, begins with, etc.) on those possible values that can be accepted by every class attribute.

Data filters can be defined and used exceptionally, otherwise they can be stored by the operator and then recalled (by the same operator or by operators of other user groups, which get the

permission to use them by the system Administrator)

See also: Class, View

6.1.12. GIS

A GIS is a system able to produce, manage and analyse spatial data by associating geographic elements to one or more alphanumeric descriptions.

GIS functionalities in CMDBuild allow you to create geometric attributes (in addition to standard attributes) that represent, on plans / maps, markers position (assets), polylines (cable lines) and polygons (floors, rooms, etc.).

See also: BIM

6.1.13. GUI FRAMEWORK

It is a user interface you can completely customise. It is advised to supply a simplified access to the application. It can be issued onto any webportals and can be used with CMDBuild through the standard REST webservice.

See also: Mobile, Webservice

6.1.14. ITIL

"Best practices" system that established a "standard de facto"; it is a nonproprietary system for the management of IT services, following a process-oriented schema (Information Technology Infrastructure Library).

ITIL processes include: Service Support, Incident Management, Problem Management, Change Management, Configuration Management and Release Management.

For each process, ITIL handles description, basic components, criteria and tools for quality management, roles and responsibilities of the resources involved, integration points with other processes (to avoid duplications and inefficiencies).

See also: Configuration

6.1.15. LOOKUP

The term "Lookup" refers to a pair of values (Code, Description) set by the administrator in the Administration Module.

These values are used to bind the user's choice (at the form filling time) to one of the preset values.

With the Administration Module it is possible to define new "LookUp" tables according to organization needs.

6.1.16. MOBILE

It is a user interface for mobile tools (smartphones and tablets). It is implemented as multi-platform app (iOS, Android) and can be used with the CMDDB through the REST webservice.

See also: GUI Framework, Webservice

6.1.17. PROCESS

The term "process" (or workflow) refers to a sequence of steps that realize an action.

Each process will take place on specific assets and will be performed by specific users.

A process is activated by starting a new process (filling related form) and ends when the last

workflow step is executed.

See also: Workflow step

6.1.18. RELATION

A relation is a link between two CMDBuild cards or, in other words, an instance of a given domain.

A relation is defined by a pair of unique card identifiers, a domain and attributes (if any).

CMDBuild allows users, through the Management Module, to define new relations among the cards stored in the database.

See also: Class, Domain

6.1.19. REPORT

The term refers to a document (PDF or CSV) containing information extracted from one or more classes and related domains.

CMDBuild users run reports by using the Management Module; reports definitions are stored in the database.

See also: Class, Domain, Database

6.1.20. CARD

The term "card" refers to an element stored in a class.

A card is defined by a set of values, i.e. the attributes defined for its class.

CMDBuild users, through the Management Module, are able to store new cards and update / delete existing ones.

Card information is stored in the database and, more exactly, in the table/columns created for that class (Administration Module).

See also: Class, Attribute

6.1.21. SUPERCLASS

A superclass is an abstract class used to define attributes shared between classes. From the abstract class you can derive real classes that contain data and include both shared attributes (specified in the superclass) and specific subclass attributes.

For example, you can define the superclass "Computer" with some basic attributes (RAM, HD, etc.) and then define derived subclasses "Desktop", "Notebook", "Server", each one with some specific attributes.

See also: Class, Attribute

6.1.22. ATTRIBUTE TYPE

Each attribute has a data type that represents attribute information and management.

The attribute type is defined using the Administration Module and can be modified within some limitations, depending on the data already stored in the system.

CMDBuild manages the following attribute types: "Boolean", "Date", "Decimal", "Double", "Inet" (IP address), "Integer", "Lookup" (lists set in "Settings" / "LookUp"), "Reference" (foreign key), "String", "Text", "Timestamp".

See also: Attribute

6.1.23. VIEW

A view not only includes the whole content of a CMDB class, it is a group of cards defined in a logical way.

In particular, a view can be defined in CMDBuild by applying a filter to a class (so it will contain a reduced set of the same rows) or specifying an SQL function which extracts attributes from one or more related classes.

The first view type maintains all functionalities available for a class, the second one allows the sole display and search with fast filter.

See also: Class, Filter

6.1.24. WEBSERVICE

A webservice is an interface that describes a collection of methods, available over a network and working using XML messages.

With webservices, an application allows other applications to interact with its methods.

CMDBuild includes a SOAP and a REST webservice.

6.1.25. WIDGET

A widget is a component of a GUI that improves user interaction with the application.

CMDBuild uses widgets (presented as "buttons") that can be placed on cards or processes. The buttons open popup windows that allow you to insert additional information, and then display the output of the selected function.