Version

3.4

CMDBuild®

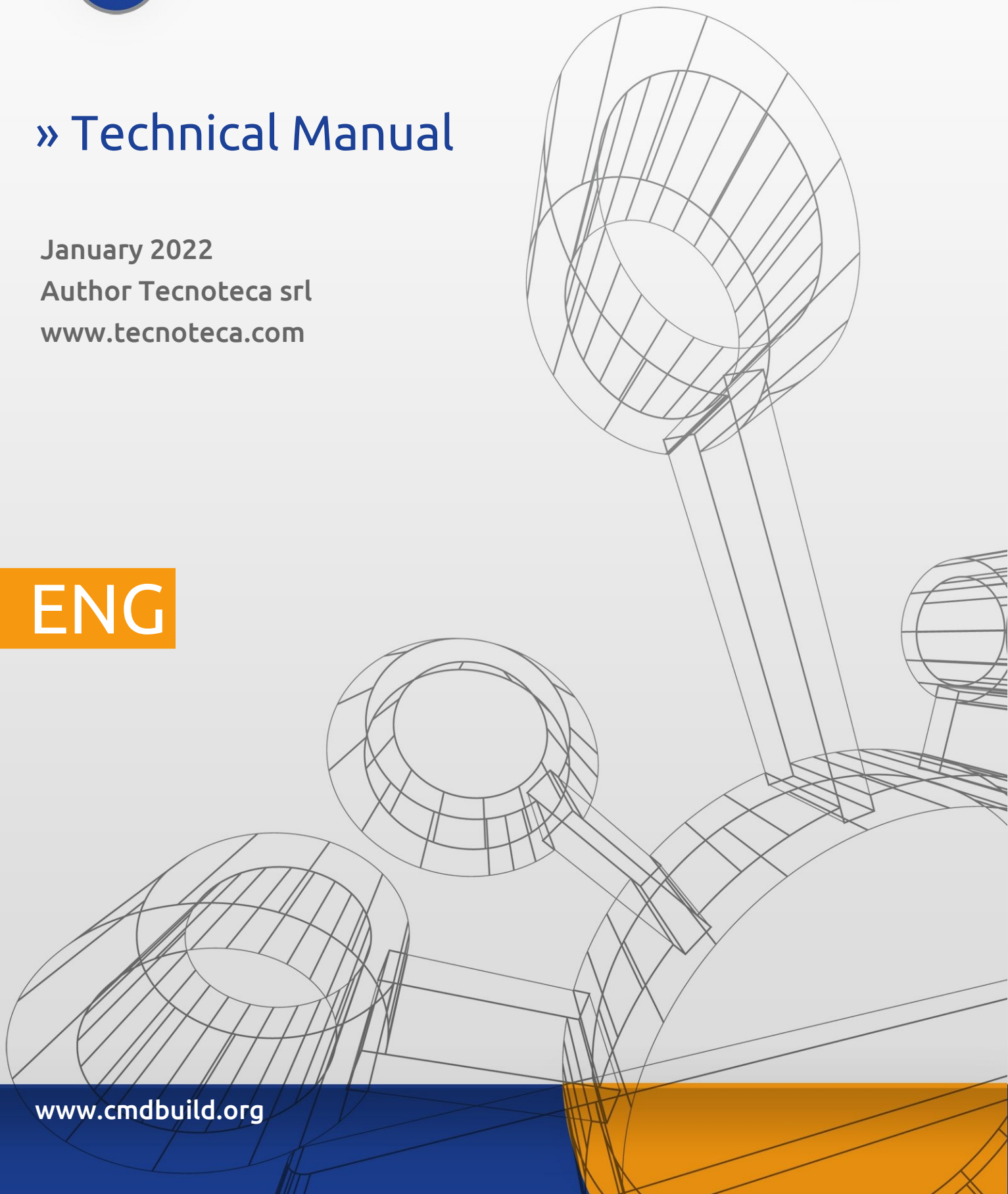
» Technical Manual

January 2022

Author Tecnoteca srl

www.tecnoteca.com

ENG



CMDBuild ® uses many great technologies from the open source community:
PostgreSQL, Apache, Tomcat, Eclipse, Ext JS, JasperSoft, JasperReports, IReport, 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.

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

Contents

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 automatism.

In a similar way, using external visual editors, the administrator can design various reports on CMDB data (printouts, graphs, bar code 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 various controls on the 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.

There is also a Scheduling, which can be supplied both automatically when filling in a data card and manually. This Scheduling will manage single or recurring deadlines related, for example, to certifications, warranties, contracts with customers and suppliers, administrative procedures, etc.

Moreover, you can use GIS features to geo reference 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 web service, 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 license 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 web service

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 web service (supplied with non-open source license 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 web service 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 for those who need certain first introductory information on CMDBuild and who are interested in knowing the general philosophy of the project.

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

- system overview ("Overview Manual")
- system usage for operators (“User Manual”)
- system administration ("Administrator Manual")
- workflow configuration (“Workflow Manual”)
- webservice details and configuration (“Webservice Manual”)

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. System configuration

In the following paragraphs the software requirements needed by the CMDBuild system and how to install and configure its components will be presented.

When planning the system configuration, information security issues must be taken into account. The activation of a web application like CMDBuild demands the availability of hardware and network components with suitable levels of security. This is required in order to avoid unwanted external accesses (firewall, DMZ) and to deliver a good system on-line availability and suitable response times.

2.1. Hardware requirements

For the CMDBuild installation a physical or virtual server is required, with the following characteristics:

- recent generation CPU
- a minimum of 8 GB of RAM, 16 GB if there will be an intensive usage of additional functionalities, such as DMS, map server, BIM server.
- minimal disk storage 120 GB, it should be much higher if you need to manage extensive archives of documents (fed by the management of the attachments).

We also advise that:

- the disk storage should be in RAID configuration
- the CMDBuild system data should be backed up daily
- an available UPS in order to avoid sudden electric power failures

2.2. Software requirements

CMDBuild installation and use require the following software components.

Operating system

Every operative system that supports JVM and the needed software will support CMDBuild.

Linux operating system is suggested because CMDBuild is more extensively tested on it.

Database

PostgreSQL from version 10 to version 12 (recommended version) is required.

You should check whether "plpgsql" is active and whether the database is set with an UTF-8 encoding.

CMDBuild uses the library "tomcat-dbcp" to connect to the database, this library is distributed with Tomcat but is not included in some Linux distributions. In such cases the library can be found in the official Tomcat distribution or in the extras/tomcat-libs/{Tomcat version dir} folder inside the CMDBuild zip file; the library must be placed in /usr/share/{Tomcat version dir}/lib.

CMDBuild supports only the PostgreSQL database, because it is the only one that implements the functionality of "derivation" of tables in the "object oriented" meaning. This is used for managing

the sub classes and for managing the history of cards.

If you use the GIS features of CMDBuild you have to also install the PostGIS spatial extension for PostGRES. If you require support installing and configuring PostGIS refer to postgis official docs and support: http://www.postgis.net/install

Servlet Container / Web Server

CMDBuild needs Apache Tomcat 8.5 or more recent (suggested Tomcat 9.0.30).

If CMDBuild is installed via GUI a fresh tomcat installation will be provided.

In order to support the UTF-8 characters when using attachments (see Installation of DMS system), edit the configuration file “server.xml” and specify the attribute URIEncoding="UTF-8" for the main "Connector" element.

You can use the web server Apache 2.2 in order to access many CMDBuild instances through virtual hosts supporting different domains.

Reference Web site for both: http://www.apache.org/

Document Management System (DMS)

Any DMS that supports the CMIS protocol can be installed for the Document Management System functionalities (that are optional).

Alfresco is the suggested DMS.

Reference website: http://www.alfresco.com/

Java Libraries

The Java Libraries are required by Apache Tomcat.

CMDBuild requires java 17.

Reference website: http://www.oracle.com/

Libraries included in the release

The CMDBuild file downloadable from the project website contains some libraries already inside the installation package, namely:

- the library for the JDBC connection to the PostgreSQL database
- the Jasper Reports libraries for the production of reports (http://www.jasperforge.org/)
- TWS Together Workflow Server 4.4 libraries which implement the workflow engine used by CMDBuild (http://www.together.at/prod/workflow/tws)
- the webservice available from the DMS Alfresco system in order to use its repository (http://www.alfresco.com/)
- the ExtJS libraries for the generation of the Ajax user interface (http://extjs.com/)
- the server and client components for the publication of geo referenced cartography (http://geoserver.org/ e http://openlayers.org/)
- the server and client components for the visualization of BIM components

In order to design custom reports, you can use the visual editor Jasper Studio (6.x) or IReport (version 5.x), which produce their descriptor in compatible format with the Jasper Reports engine

(http://jasperforge.org/projects/ireport).

For designing personalized workflows we suggest using the visual editor TWE Together Workflow Editor 4.4 (http://www.together.at/prod/workflow/twe). The editor produces in output an XPDL 2.0 file compatible with the Together Workflow Server 4.4 engine.

For integrating systems of automatic inventory we suggest using the OCS Inventory version 1.3.3 (http://www.ocsinventory-ng.org/).

Some functionalities of CMDBuild can be integrated as portlets within systems compatible with Portal JSR, among them Liferay version 6.0.6 or more recent (http://www.liferay.com/).

All software listed above are released with an Open Source license (the operating system is not included if you choose to not use the Linux operating system).

2.3. Client requirements

Every interaction made with CMDBuild (using, administrating, updating the system and structuring the database) is made via web application.

This means that the user, in order to interact with the system, will be required to have a recent generation web browser installed (supported browser: Mozilla Firefox, Google Chrome, Microsoft Explorer 11, Microsoft Edge, Apple Safari).

The web architecture ensures complete usability to any IT organization that operates in multiple locations (ie collaborative workflow); any entrusted client can connect and interact with the system using a standard web browser.

3. Installing CMDBuild

3.1. Getting started

CMDBuild can be installed in different ways, the installation via GUI is suggested. Anyhow the following list of software are required before the installation:

- PostgreSQL database (it must be started and accessible)
- Tomcat application server (only if manual installation is performed)
- the DMS Alfresco (or others supporting the CMIS protocol, if you intend to use the management of attached documents)
- the Java environment

As a first step is therefore necessary to ensure downloading and installing these products, retrieving them from the links mentioned in the previous chapter.

Warning: you must to be careful to use directories not containing spaces within the entire path.

3.2. CMDBuild installation

After obtaining and installing the previously required programs we can proceed with the CMDBuild installation.

The installations files are provided on the official CMDBuild website at the following link: http://www.cmdbuild.org/en/download/

A .war file is provided to proceed with both graphical or manual installation.

3.3. CMDBuild installation via GUI (Windows and Linux)

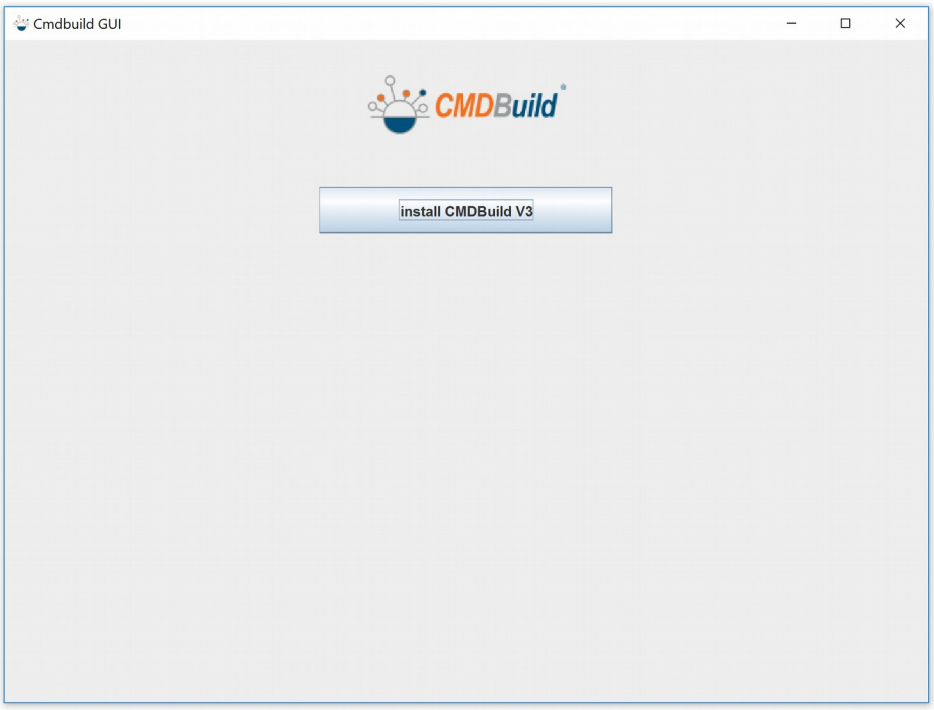
To simplify the standard installation and configuration of CMDBuild, a guided GUI installation has been provided for both windows and unix operating systems.

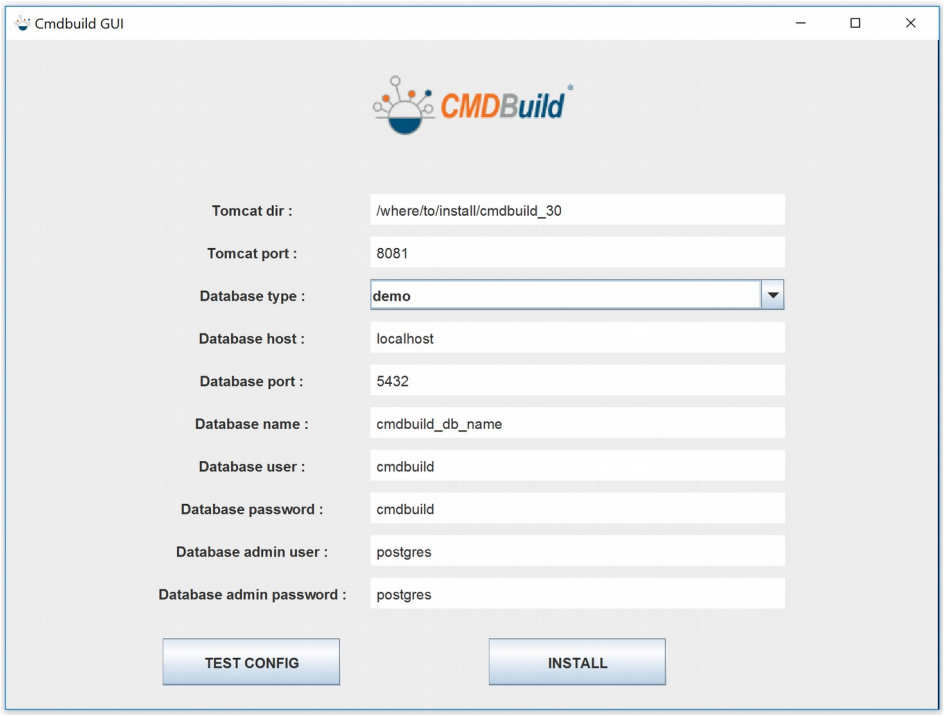
Once having downloaded the .war file open a terminal (on linux) or a cmd window (on windows) and move to the folder containing the .war file. To run the GUI type the following command:

java -jar cmdbuild.war -v

This command will launch the graphical interface like in the following screenshot:

By clicking “install CMDBuild V3” we’ll be taken to the first configuration steps:

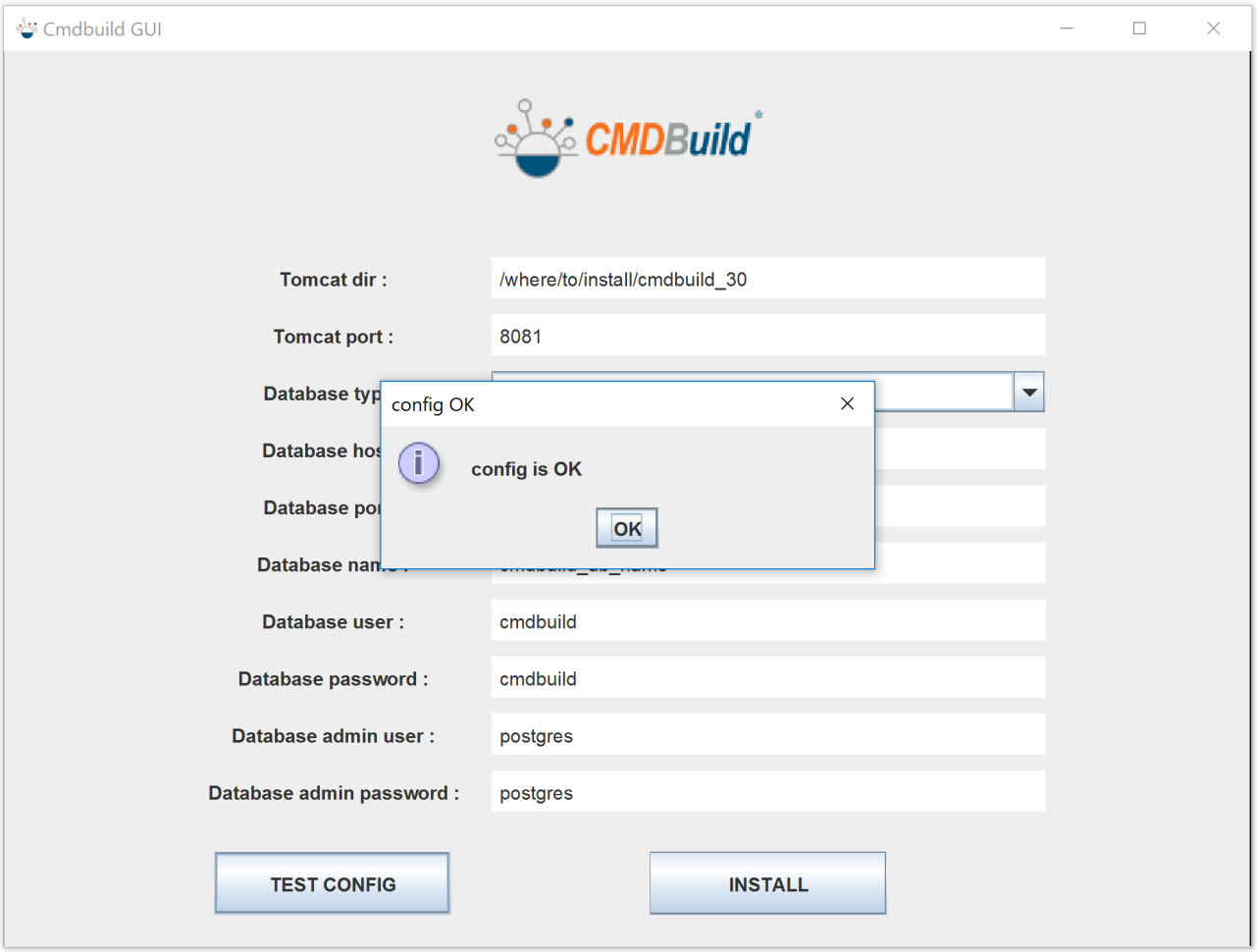

Cmdbuild GUI
CMDBuild
install CMDBuild V3


Cmdbuild GUI
CMDBuild
Tomcat dir :
/where/to/install/cmdbuild_30
Tomcat port :
8081
Database type :
demo
Database host :
localhost
Database port :
5432
Database name :
cmdbuild_db_name
Database user :
cmdbuild
Database password :
cmdbuild
Database admin user :
postgres
Database admin password :
postgres
TEST CONFIG
INSTALL

In this form we can decide the basic configuration of CMDBuild, the various field are now described:

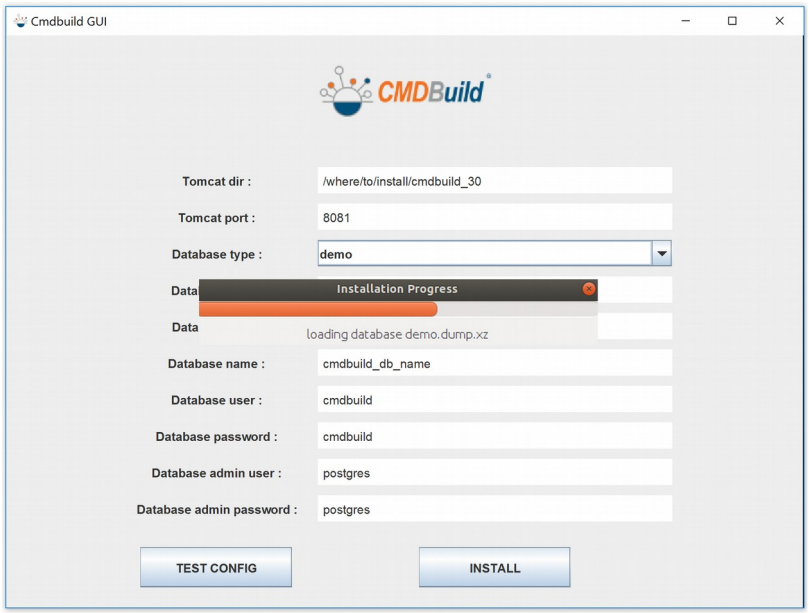
- Tomcat dir: the path where to install a fresh tomcat
- Tomcat port: the port that tomcat will use to communicate
- Database type: two basic databases are presented, an empty one where only the basic structure is presented but no data is added, or a demo one where other than the basic structure some data are added in for demonstration purposes
- Database host: the host where to install tomcat, localhost if we want to run it on our machine
- Database port: the port used to communicate to the database
- Database name: the name of the database that will be created
- Database user: the basic user of the database, default: cmdbuild
- Database password: the basic user password of the database, default: cmdbuild
- Database admin user: the admin user of the database, default: postgres
- Database admin password: the admin user password of the database, default: postgres

A good practice is to test the configuration with the button “Test Config”, if everything is correctly configured the following pop-up will appear:

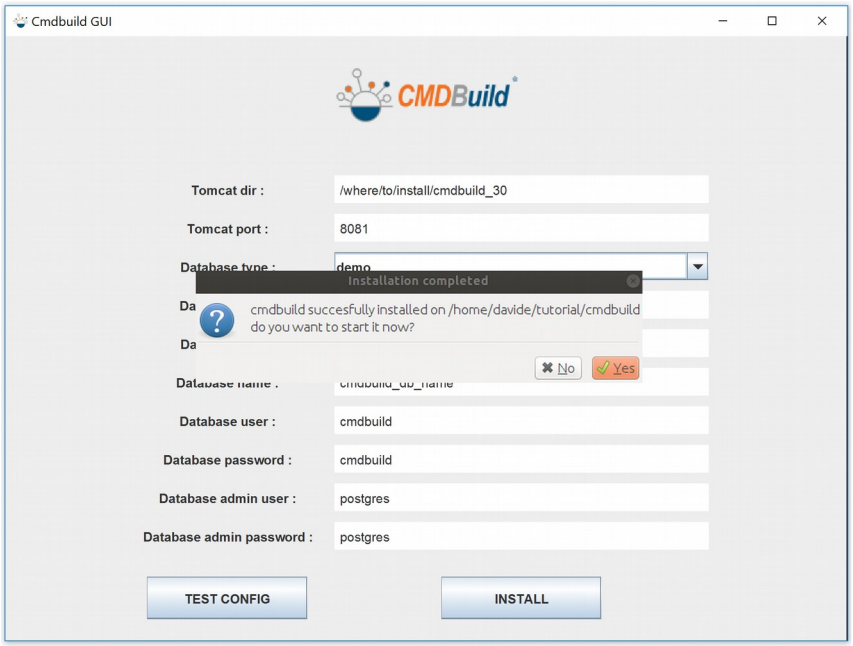


We can now proceed with the “INSTALL” button.

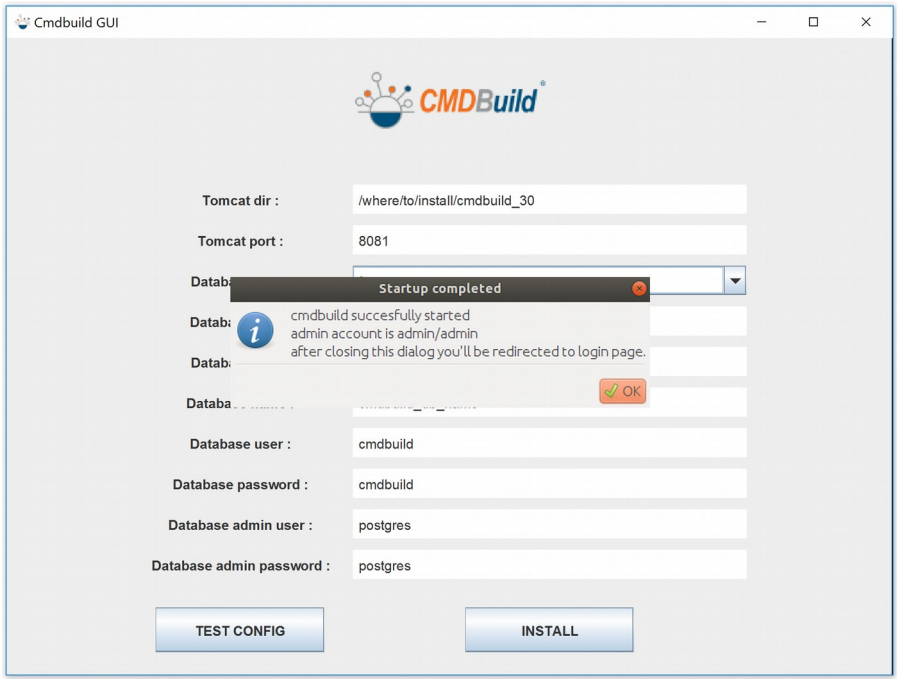
Apache tomcat and CMDBuild will be installed, also the selected database will be loaded:

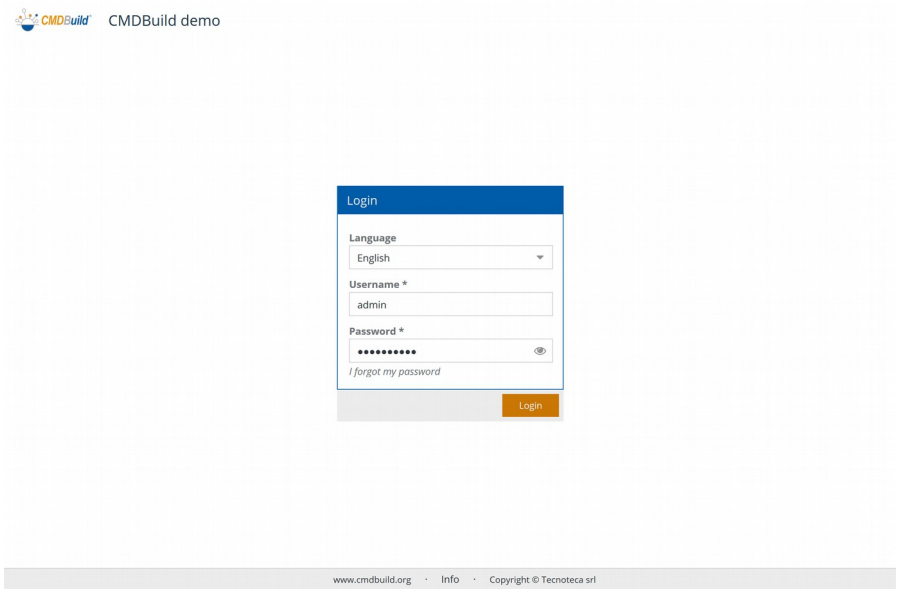


As soon as the installation will be completed a pop-up will ask us if we want to start the application or not:



By choosing yes Apache tomcat will load and the installer will notify us as soon as the application is ready to be used:





3.4. CMDBuild manual installation (Linux and Windows)

If instead of the graphical installation we want to manually install and configure CMDBuild, a different procedure has to be followed. First of all Apache Tomcat (and every required software previously listed) has to be manually installed (check the chapter above for download links and supported versions). Once Tomcat is successfully installed we can proceed with the manual installation.

Note: if you are installing OpenMAINT you should replace the references to "cmdbuild" with "openmaint".

The downloaded file (cmdbuild-3.4.war) contains the web application files that can be extracted in the tomcat installation folder under /webapps/cmdbuild .

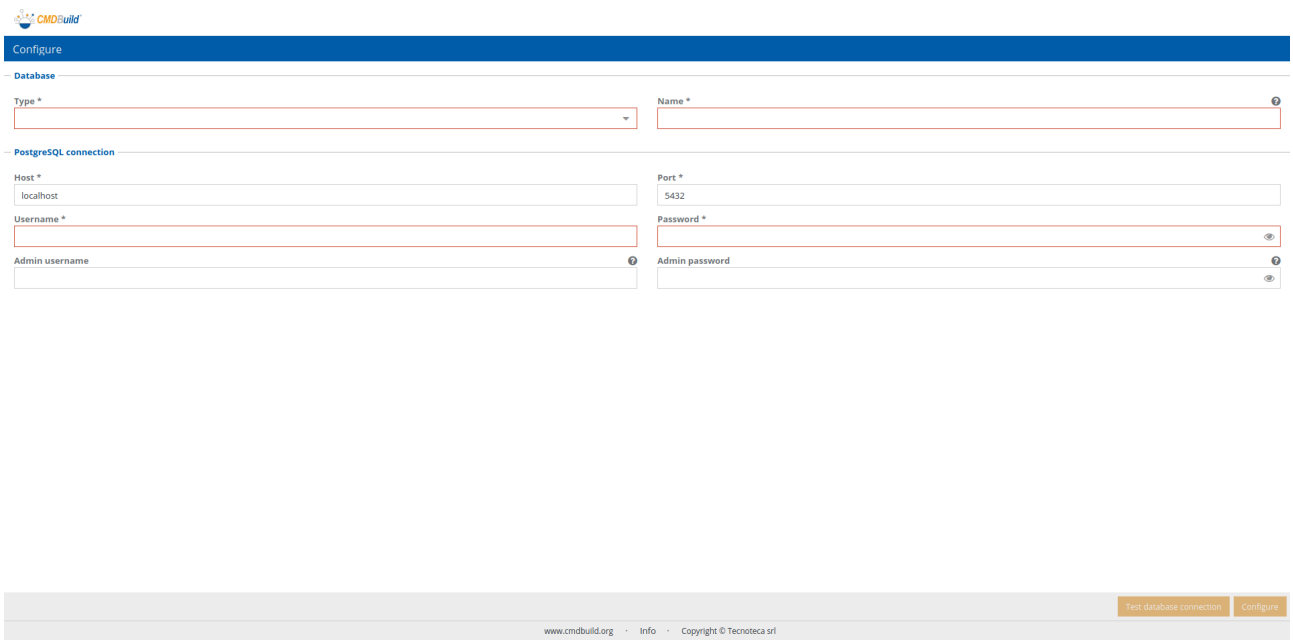
Doing this will provide us with a working CMDBuild installation, but no database has yet been created.

To configure the database we can either use the database configuration wizard provided after the first tomcat startup, or we can proceed with a manual configuration via command line.

3.5. CMDBuild database configuration via Db Config Wizard (Linux and Windows)

As in CMDBuild version 2.5 or older, a wizard to perform the database configuration has been provided for new CMDBuild installations.

After configuring and launching a tomcat instance with the cmdbuild webapp, by accessing the application via browser we will be redirected to the database configuration page:



In this form we can configure the database with the following fields:

- Type: the database we want to configure, either one of the provided ones (demo, empty), an existing one or from a file

- Name: the name used by postgres
- Host: postgres host
- Port: postgres port
- Username: standard postgres username (suggested cmdbuild)
- Password: standard postgres password (suggested cmdbuild)
- Admin username: administrator postgres username (suggested postgres)
- Admin password: administrator postgres password (suggested postgres)

Once every field has been compiled it is suggested to use the “Test database connection” to verify the validity of the configuration, if no errors are displayed we can proceed by pressing the configure button.

After a successful configuration a list of patches that have to be applied to the uploaded database might be shown, by pressing “Apply patches” the system will proceed with the application.

As soon as the login page is displayed the system is ready to be used.

3.6. CMDBuild manual database configuration for Windows

If the OS in use is Windows, the database import has to be done manually either via software like pgAdmin or via postgres command line, the dump file for the database is provided under WEB-INF/sql/dump located in the webapp folder. If you want to load a dump which includes gis/postgis features, refer to postgis docs for any issue regarding postgis configuration and/or postgis version upgrade/migration.

Example of postgis installation with postgres 12 and postgis 3.x for Windows 10:

-Download and Install postgres 12 from the official website http://postgresql.org/download/windows

-Download and install postgis for postgres 12 from the official online repository: http://download.osgeo.org/postgis/windows/pg12

Connect to the postgres database by using a software like pgadmin 4. As the default administration user create a new user named cmdbuild with password. Next create a database for the user cmdbuild, you can now reconnect to this database as user cmdbuild and create a gis schema:

```
create schema gis;
```

Ceate the postgis extension (note: you may have to grant superuser permissions to cmdbuild user for this command; you should revoke superuser permission after):

```
create extension postgis schema gis;
```

You can now proceed with importing a database. We can use, for example, one of the dumps provided with the CMDBuild installation.

By using pgadmin as user cmdbuild, we can perform a restore with cmdbuild as target database, the import will generate various warnings that can be ignored, those are caused by some duplicated information in the dump file and in the existing database (gis function and types).

After the process is completed the application can be executed by launching tomcat with the startup file, remember that if the port or any tomcat configurations have to be changed, this can be done by modifying the tomcat configuration file called server.xml in the folder <tomcat_install_folder>/conf.

3.7. CMDBuild manual database configuration for Linux

For the linux systems the command that will be used is dbconfig, and it can be called by using the cmdbuild.sh file in the webapp folder. This command can perform the creation of a new database based on a dump, and other functions described in the other documentations.

At this point the demo database provided with the CMDBuild war file can be imported. To do so you can use the dbconfig command like in the following:

bash webapps/cmdbuild/cmdbuild.sh dbconfig create demo -configfile /path/to/configfile

CMDBuild will now setup a new database with the name provided in the configuration and will load the demo dump provided in the war file.

4. CMDBuild version update

4.1. CMDBuild update

In order to update CMDBuild to a newer version follow this procedure:

- Turn off tomcat execution and backup any needed information (i.e. the database)
- Save any needed configuration from the configuration folder: <tomcat_installation_folder>/webapps/<webapp_name>/WEB-INF/conf
- Delete the directory of your CMDBuild web application in tomcat (<tomcat_installation_folder>/webapps/<webapp_name>)
- Unpack the war file of the newer version of CMDBuild in the same folder
- Clean your browser cache to make sure that everything is updated

When the application is next launched, on the login screen a pop-up notifying that there is a new version installed will appear.

To restore the previously saved configurations tomcat has to be turned off again and the old configurations have to be moved to the configuration folder.

5. Configuration to access a DMS through CMIS

5.1. Introduction

CMDBuild allows the integration with any document management system that supports CMIS, an open standard to exchange document data through the web protocol, supported by all the main products of the sector.

The acceptance of such protocol allows to support base features to:

- Attach any kind of file to any data card
- Assign a category to every file
- Display the uploaded files

The management of categories and metadata at the moment is supported only for the protocol CMIS 1.1, but CMDBuild supports even DMS based on CMIS 1.0. In case of the need for categories and metadata a DMS based on CMIS 1.1 has to be used.

5.2. Configuration of categories management

In this section you will learn how to configure a DMS that can support the protocol CMIS 1.1, in order to manage the classification of attachments when updating and refreshing them on CMDBuild cards.

Here's an example of the configuration for Alfresco 5.0.

Note that the names used here are unbinding.

To setup some custom categories with alfresco, two files have to be created and saved in the alfresco folder:

${ALFRESCO_HOME}/tomcat/shared/classes/alfresco/extension

1) cmdbuild-model-context.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org/dtd/spring-beans.dtd'>
<beans>
   <!-- Registration of new models -->
         <bean id="example.dictionaryBootstrap" parent="dictionaryModelBootstrap" depends-on="dictionaryBootstrap">
      <property name="models">
         <list>
            <value>alfresco/extension/cmdbuild-model.xml</value>
         </list>
      </property>
   </bean>
</beans>
```

2) cmdbuild-model.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Definition of new Model -->
<!-- The important part here is the name - Note: the use of the my: namespace
    which is defined further on in the document -->
<model name="cmdbuild:module" xmlns="http://www.alfresco.org/model/dictionary/1.0">
  <!-- Optional meta-data about the model -->
  <description>Custom Model for CMDBuild</description>
  <author>CMDBuild Team</author>
  <version>1.0</version>
  <!-- Imports are required to allow references to definitions in other models -->
  <imports>
        <!-- Import Alfresco Dictionary Definitions -->
    <import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d"/>
    <!-- Import Alfresco Content Domain Model Definitions -->
    <import uri="http://www.alfresco.org/model/content/1.0" prefix="cm"/>
  </imports>
  <!-- Introduction of new namespaces defined by this model -->
   <!-- NOTE: The following namespace my.new.model should be changed to reflect your own
namespace -->
  <namespaces>
    <namespace uri="org.cmdbuild.dms.alfresco" prefix="cmdbuild"/>
  </namespaces>
   <aspects>
    <!-- Definition of new Content Aspect: Image Classification -->
    <aspect name="cmdbuild:classifiable">
      <title>Classfication</title>
      <properties>
        <property name="cmdbuild:classification">
          <type>d:text</type>
        </property>
      </properties>
    </aspect>
  </aspects>
</model>
```

5.3. Configuration of the DMS in CMDBuild

Via REST commands the configurations of the DMS on CMDBuild can be changed, either via editconfig rest command to open a gui showing all the available configurations (if CMDBuild is on the local machine) or via setconfig/setconfigs rest command, in example:

cmdbuild.sh restws setconfig org.cmdbuild.dms.enabled true

General configs:

Config	Type	Description
org.cmdbuild.dms.enabled	Boolean	Enables or disables the DMS service
org.cmdbuild.dms.service.type	String	DMS service type (cmis, postgres, sharepoint_online); cmis is a standard protocol used, for example, by Alfresco dms; postgres is an embedded dms implementation that relies upon cmdbuild postgres db
org.cmdbuild.dms.category.lookup	String	Sets the base lookup DMS category
org.cmdbuild.dms.regularAttachments.maxFileSize	Int	Maximum allowed file size, expressed in MB
org.cmdbuild.dms.incomingEmailAttachments.allowedFileExtensions	List<String>	Allowed file extensions, lowercase, for incoming email (rejected attachments will be ignored and print a warning, without affecting email processing)
org.cmdbuild.dms.autolink.path	String	Autolink base path (replaces dms provider base path, for links)
org.cmdbuild.dms.autolink.script	String	Autolink helper script

CMIS related configs:

Config	Type	Description
org.cmdbuild.dms.service.cmis.user	String	CMIS service username
org.cmdbuild.dms.service.cmis.password	String	CMIS service password
org.cmdbuild.dms.service.cmis.path	String	CMIS service path
org.cmdbuild.dms.service.cmis.url	String	CMIS service URL

Sharepoint related configs:

Config	Type	Description
org.cmdbuild.dms.service.sharepoint.user	String	Sharepoint service username
org.cmdbuild.dms.service.sharepoint.password	String	Sharepoint service password
org.cmdbuild.dms.service.sharepoint.url	String	Sharepoint service URL
org.cmdbuild.dms.service.sharepoint.path	String	Sharepoint service path

org.cmdbuild.dms.service.sharepoint.graphApi.url	String	Sharepoint graph api service URL
org.cmdbuild.dms.service.sharepoint.auth.serviceUrl	String	Sharepoint authentication service url
org.cmdbuild.dms.service.sharepoint.auth.clientId	String	Sharepoint authentication client Id
org.cmdbuild.dms.service.sharepoint.auth.clientSecret	String	Sharepoint authentication client secret
org.cmdbuild.dms.service.sharepoint.auth.protocol	String	Sharepoint authentication protocol
org.cmdbuild.dms.service.sharepoint.auth.resourceId	String	Sharepoint authentication resource Id
org.cmdbuild.dms.service.sharepoint.auth.tenantId	String	Sharepoint authentication tenant Id
org.cmdbuild.dms.service.sharepoint.model.authorColumn	String	Sharepoint model author mapping column
org.cmdbuild.dms.service.sharepoint.model.categoryColumn	String	Sharepoint model category mapping column
org.cmdbuild.dms.service.sharepoint.model.descriptionColumn	String	Sharepoint model description mapping column

6. Backup of CMDBuild data

6.1. Database backup

In CMDBuild every information is stored in the PostgreSQL Database, so in order to perform a backup of the information we need to backup the database.

Various ways can perform a backup of the database, programs like PgAdmin can perform this action with the support of a graphical interface. Otherwise PostgreSQL provides a function to perform a database backup.

In example:

```
pg_dump --host localhost --port 5432 --username “postgres” --format custom --verbose --file /backup/cmdbuild/cmdbuild.backup cmdbuild_db
```

This command will create a backup of the database with the name “cmdbuild.backup”. This file will be the backup of the database “cmdbuild_db” on the PostgreSQL installed on localhost at port 5432, performing the authentication as the user “postgres”, compressing the database with a custom format and notifying the user with what the program is executing (with the verbose parameter).

The location of where the backup file will be saved is /backup/cmdbuild.

6.2. Database restore

A backup can be restored at any time with a simple procedure.

CMDBuild provides different functions to perform various actions, such as database related actions.

If the purpose is to restore a backup this command will restore the previously saved backup by dropping the current database and creating a new one:

bash webapps/cmdbuild/cmdbuild.sh dbconfig recreate /path/to/database/backup.backup

7. Backup of Alfresco data

To backup data in Alfresco you just have to backup the PostgreSQL database (hereafter we will assume that it is in the database PostgreSQL).

As for CMDBuild, the command to backup the database on command line is as follows:

```
pg_dump --host localhost --port 5433 --username "postgres" --format custom --verbose --file /backup/alfresco/alfresco.backup alfresco_db
```

It is also necessary backup - as zip or tar.gz file - also the whole directory containing the repository where files are saved (e.g.: C:/alfresco/repository for Windows systems or /var/alfresco/repository for *nix systems).

```
#!/bin/sh
ALFRESCO_SERVICE="/etc/init.d/alfresco"

TT_PG_BACKUP="/usr/local/bin/tt_pg_backup.sh"

BACKUP_HOME="/backup"
ALFRESCO_BACKUP_HOME="${BACKUP_HOME}/alfresco"
BACKUP_LOG="/var/log/cmdbuild/crontab-backup.log 2>&1"

ALFRESCO_DATABASE="alfresco"

ALFRESCO_DATA="/var/lib/alfresco/data"
TIMESTAMP=`date +%Y%m%d%H%M%S`
ALFRESCO_DATA_BACKUP="alfresco-data-${TIMESTAMP}.tar.gz"

logger "Stopping Alfresco service"
${ALFRESCO_SERVICE} stop
sleep 5

logger "Backupping Alfresco database"
${TT_PG_BACKUP} "${ALFRESCO_BACKUP_HOME}" "${ALFRESCO_DATABASE}" > ${BACKUP_LOG}

logger "Backupping Alfresco data"
tar czf "${ALFRESCO_BACKUP_HOME}/${ALFRESCO_DATA_BACKUP}" ${ALFRESCO_DATA}
logger "Alfresco database/data have been backed up."
logger "Starting Alfresco service"
${ALFRESCO_SERVICE} start
```

7.1. Backup schedule

To schedule a periodical data backup on Linux systems, we suggest you to go on as follows:

1. create a /etc/cron.d/backup file
2. insert the commands into the backup file, with the proper cron syntax.

For a more efficient cleaning we recommend you to create various single scripts, one for each system, and call them in this file rather than write the commands directly into the cron file itself.

An example:

```
00 19 * * *     root   /usr/local/bin/tt_pg_backup.sh /backup/cmdbuild cmdbuild &>> /var/log/cmdbuild/backup/backup.log

10   19   *   *   7            root      /usr/local/bin/alfresco-backup.sh   &>> /var/log/cmdbuild/backup/backup.log
```

8. Authentication modes

8.1. Introduction

Thanks to proper configurations of CMDBuild, the authentication control can be delegated to external services.

This possibility concerns the control of the account (username and password). Profiles and permissions will still be managed withing the CMDBuild group to which the user belongs.

Via REST commands the configurations of the authentication can be changed, either via editconfig rest command to open a gui showing all the available configurations (if CMDBuild is on the local machine) or via setconfig/setconfigs rest command, in example:

cmdbuild.sh restws setconfig org.cmdbuild.auth.case.insensitive true

8.2. Configuration of the authentication type

CMDBuild supports the following authentication methods:

- Default authentication
- SSO CAS authentication
- SSO SAML authentication
- Oauth2 authentication

The default authentication can be configured to utilize two different user repository:

- DB stored credentials
- LDAP system

In addition there are the following additional authentication methods:

- RSA authentication
- Header authentication
- Custom login (similar to header auth but with a control script addition)

To configure which type or types of authentication to use, from version 3.4, it is possible to specify different auth modules with the following configuration schema

Config	Type	Description
org.cmdbuild.auth.modules	String	List of auth module names
org.cmdbuild.auth.modules.{ModuleName}.type	String	Authentication type for the specified {ModuleName}
org.cmdbuild.auth.modules.{ModuleName}.description	String	Auth module description, shown in auth login button
org.cmdbuild.auth.modules.{ModuleName}.enabled	Boolean	Specifies if the module is enabled or not
org.cmdbuild.auth.modules.{ModuleName}.hidden	Boolean	Specifies if the module is hidden or not, if hidden it’s only used by passing cm_login_module in the request

org.cmdbuild.auth.modules.{ModuleName}....	String	Additional configurations for the specified {ModuleName}

org.cmdbuild.auth.modules.{ModuleName}.type can have the following values as specified above:

- Default authentication
- SSO CAS authentication
- SSO SAML authentication
- Oauth2 authentication

CAS related configs:

Config	Type	Description
org.cmdbuild.auth.module.cas.server.url	String	CAS service URL
org.cmdbuild.auth.module.cas.login.page	String	CAS url login page
org.cmdbuild.auth.module.cas.service.param	String	CAS service parameter
org.cmdbuild.auth.module.cas.ticket.param	String	CAS service ticket parameter

SAML related configs:

Config	Type	Description
org.cmdbuild.auth.module.saml.handlerScript	String	SAML authentication response handler script
org.cmdbuild.auth.module.saml.idp.cert	String	SAML idp certificate
org.cmdbuild.auth.module.saml.idp.id	String	SAML idp id URL
org.cmdbuild.auth.module.saml.idp.login	String	SAML idp login URL
org.cmdbuild.auth.module.saml.idp.logout	String	SAML idp logout URL
org.cmdbuild.auth.module.saml.logout.enabled	Boolean	SAML idp logout enabled flag
org.cmdbuild.auth.module.saml.requireSignedAssertions	Boolean	SAML require signed assertion flag
org.cmdbuild.auth.module.saml.requireSignedMessages	Boolean	SAML require signed messages flag
org.cmdbuild.auth.module.saml.signatureAlgorithm	String	SAML signature algorithm URL
org.cmdbuild.auth.module.saml.sp.baseUrl	String	SAML service provider base URL
org.cmdbuild.auth.module.saml.sp.cert	String	SAML service provider base certificate
org.cmdbuild.auth.module.saml.sp.id	String	SAML service provider id URL
org.cmdbuild.auth.module.saml.sp.key	String	SAML service provider private key
org.cmdbuild.auth.module.saml.strict	String	SAML service provider strict validation flag

Oauth related configs:

Config	Type	Description
org.cmdbuild.auth.module.oauth.clientId	String	OAuth client id
org.cmdbuild.auth.module.oauth.clientSecret	String	OAuth client secret
org.cmdbuild.auth.module.oauth.login.attr	String	OAuth login attribute to be matched with cmdbuild users
org.cmdbuild.auth.module.oauth.login.type	String	OAuth login type matching (username or email)
org.cmdbuild.auth.module.oauth.logout.enabled	Boolean	OAuth logout enabled flag
org.cmdbuild.auth.module.oauth.logout.redirectUrl	String	OAuth logout redirect URL
org.cmdbuild.auth.module.oauth.protocol	String	OAuth protocol (i.e. msazureoauth2)
org.cmdbuild.auth.module.oauth.redirectUrl	String	OAuth local url accepted from the provider
org.cmdbuild.auth.module.oauth.resourceId	String	OAuth resource id
org.cmdbuild.auth.module.oauth.scope	String	OAuth scope
org.cmdbuild.auth.module.oauth.serviceUrl	String	OAuth service URL
org.cmdbuild.auth.module.oauth.tenantId	String	OAuth tenant id

LDAP related configs:

Config	Type	Description
org.cmdbuild.auth.ldap.server.address	String	LDAP server address URL
org.cmdbuild.auth.ldap.server.url	String	LDAP server url (if set, will override server host, port and ssl config); you may specify multiple urls separated by one space
org.cmdbuild.auth.ldap.server.port	Integer	LDAP server port
org.cmdbuild.auth.ldap.basedn	String	LDAP base dn for user query
org.cmdbuild.auth.ldap.bind.attribute	String	LDAP user bind attribute
org.cmdbuild.auth.ldap.followReferrals	Boolean	LDAP follow referrals enable flag
org.cmdbuild.auth.ldap.search.auth.method	String	LDAP auth method (none, simple, strong)
org.cmdbuild.auth.ldap.search.auth.password	String	LDAP auth password
org.cmdbuild.auth.ldap.search.auth.principal	String	LDAP auth principal
org.cmdbuild.auth.ldap.search.filter	String	LDAP search filter
org.cmdbuild.auth.ldap.use.ssl	Boolean	LDAP ssl enable flag
org.cmdbuild.auth.ldap.use.tls	Boolean	LDAP tls enable flag

General configs:

Config	Type	Description
org.cmdbuild.auth.case.insensitive	Boolean	Case insensitive login flag
org.cmdbuild.auth.loginAttributeMode	String	Login attribute mode (username, email, auto_detect_email)
org.cmdbuild.auth.loginServiceReturnSessionId	String	Return session id at login (auto, always)
org.cmdbuild.auth.logoutRedirect	String	Logout redirect URL
org.cmdbuild.auth.maxLoginAttempts.count	Integer	Max login attempts count
org.cmdbuild.auth.maxLoginAttempts.window	Integer	Time in seconds between max login attempts
org.cmdbuild.auth.preferredPasswordAlgorythm	String	Password encryption algorithm (legacy, cm3easy, cm3)
org.cmdbuild.auth.users.expireInactiveAfterPeriod	Iso 8601	Period of time for inactive users expiry

8.3. Configuring LDAP authentication

This section documents how to configure authentication within CMDBuild via LDAP.

In order to manage the user permissions within CMDBuild is necessary that users that have to access to CMDBuild they are also present within the webapp.

For example, if a user with LDAP UID j.doe needs accessing CMDBuild as a user of the "technicians" group, you have to perform these steps:

- user creation in j.doe CMDBuild with a default password (not necessarily that of LDAP)
- creation of the Technical Group and definition of the relevant permits
- adding user to group j.doe Technicians

At this point, when you authenticate j.doe, his credentials will be verified (using the authentication chain defined in auth.methods) against the LDAP tree.

8.4. Single sign on configuration through CAS

This section documents how to configure the single sign on (SSO) within CMDBuild via CAS.

The authentication runs as follows:

- the user asks for the CMDBuild url
- the CAS authenticator sends the request to the CAS server (${cas.server.url} + ${cas.login.page}) specifying the CMDBuild access url (in the ${cas.service.param} paramter)
- the CAS server answers with a ticket (${cas.ticket.param} paramter) which you can extract the username from
- if the username is properly validated/extracted, then CMDBuild gets on with the login

In order to manage the user permissions within CMDBuild is necessary that users that have to access to CMDBuild they are also present within the webapp.

9. Mobile interface activation

9.1. Introduction

CMDBuild “mobile” is an interface that can be used on smartphones and tablets in order to run useful application features during the activities on field.

It is created with “Sencha Touch” (a Javascript framework developed by Sencha, the same productor of Ext JS framework used by the CMDBuild desktop interface), and it is able to interact with CMDBuild via REST webservice.

CMDBuild mobile implements the main features of the desktop interface: multilingual, multi group login, navigation menus, class management, relations, attachments, …

9.2. Components and architecture

The application is created with the usage of the following components:

- Sencha Touch (Javascript framework created by Sencha)
- Cordova (mobile cross-platform framework)
- Deft JS (mobile enterprise extension for applications developed with Sencha Touch)
- log4javascript (javascript logging framework)
- Crosswalk (tool for the application deployment on a custom webview independent from the android version)
- Siesta (library for the usage of unit/integration tests)

On the server side, the REST web service layer is developed using the Apache CXF framework, that is already integrated in CMDBuild, also used for the SOAP web service layer.

The system retraces the software architecture of the REST web service with the following features:

- features divided in web resources
- addressable resources (URI)
- HTTP standard methods (GET, POST, PUT, DELETE)
- JSON media type
- link for the resource navigation
- stateless

9.3. Compatibility

The mobile system is compatible with:

- Android 4.0.3 or more recent
- IOS 6 or more recent

9.4. Limitation of use

The mobile interface is available only with the non-open source license, which allows only those who have signed with Tecnoteca srl a maintenance service for the CMDBuild application, and only until the maintenance is active, with a limited additional charge.

10. GUI Framework activation

10.1. Introduction

The GUI Framework makes available a simplified interface to non-technical staff.

The GUI Framework includes the following main features:

- it can be activated in portals based on different technologies as it is developed in javascript / JQuery environment
- it allows an (almost) unlimited freedom when projecting the graphic layout, defined through an XML descriptor and with the possibility of intervening on the CSS
- it grants a quick configuration thanks to predefined functions (communication, authentication logics, etc.) and to native graphic solutions (forms, grids, upload buttons and other widgets)
- it interacts with CMDBuild through the REST webservice
- it is able to gather data from the database of other applications, allowing the management of mix solutions

10.2. Configuration

The framework defines HTML pages starting from their definition in XML.

Such pages can be inserted into a HTML file, allowing the framework to insert into existing portals access points to CMDBuild data.

The system configurability can be achieved through the definition of custom CSS and Javascript.

The element that should be inserted into the portal html is a html container (DIV, IFRAME ...) in the following format:

```
   <script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></
script>
   <script type="text/javascript" src="http://10.0.0.107:8080/cmdbuild-gui-
framework/api/cmdbuildAP.js"></script>
   <script type="text/javascript">
   $( document ).ready(function(){
           $("#cmdbuilForm").cmdbuildAP({
               apiUrl : 'http://10.0.0.107:8080/cmdbuild/services/rest/',
               appRootUrl :
'http://10.0.0.107:8080/cmdbuild-gui-framework/api/',
               appConfigUrl :
'http://10.0.0.107:8080/cbap/cbap/config/bologna/',
               cqlUrl : 'http://10.0.0.107:8080/cbap/cbap/cql/',
               customjs : [
            'backend/GUIForm.js',
                        'backend/Process.js'
        ],
               start: "home.xml",
               theme: [
           "jquery-ui.theme-crt-toscana.min.css",
                        "custom.css"
               ],
```

```
    });
     });
</script>
<div id="cmdbuilForm"></div>
```

The style configuration is designated to CSS, which can directly uploaded from the Html tag. This makes the GUI flexible, allowing the programmer to include it into the portal so that is does not look like an unrelated element.

You can also define the servers that contain the CMDBuild system, the form configuration files and the Cql engine, the native data query language in CMDBuild.

In the tag customjs some GUI customization files are inserted.

The tag uploads the GUI engine configuring it with the above-mentioned files.

The forms are defined in XML through a language that is similar to HTML, with the possibility of defining data management forms directly linked to metadata defined in CMDBuild.

All GUI tags, commands and behaviours can be configured again by the programmer according to the system criteria.

Since the library refers to JQuery, where the GUI Framework was developed, the programmer can natively use every plugin compatible with JQuery, developing the available features.

Another element that marks the GUI operation is the "backend" mechanism, i.e. the Javascript classes that link the GUI to the servers. In this way the application provides further freedom provided by the possibility to define the data format the server is waiting for and the name of the server you want to link to.

11. GeoServer

11.1. Geo Server introduction

CMDBuild includes the ability to manage the geo-reference of assets or of any other information entities (such as customers, suppliers, locations, etc.) through the visualization on a map and/or floor plans.

The geo-reference on the territory is made by integration external services such as “OpenStreetMap”, Google maps, and more, while the management of plans has been implemented by using the GeoServer open source system.

Through the use of GeoServer you can add custom layers (e.g. plans) that you can use in the GIS module.

Supported formats are Shape, GeoTiff and WorldImage

11.2. Installing Geo Server

In order to install Geo Server it is necessary to:

- Download the application from the GeoServer official website: http://geoserver.org/
- Deploy the war file in your tomcat installation folder under webapp
- Log in to GeoServer with username admin and password geoserver
- Delete any previously installed workspace
- Create a new workspace with an arbitrary name (I.E. “cmdbuild”)

In order to print the maps from CMDBuild, an additional plugin has to be installed, the plugin can be downloaded from https://docs.geoserver.org/maintain/en/user/extensions/printing/index.html

To setup the plugin the procedure is the following:

- Download the library geoserver-2.x.x-printing-plugin.zip
- Extract the file and copy the jar files in the geoserver webapp folder under the tomcat installation folder (<tomcat_installation_folder>/webapps/geoserver/WEB-INF/lib/)
- After restarting Tomcat, a “printing” folder will be created in the data folder under the geoserver webapp
- Add the file named “config.yaml” in the printing folder, this file can be found in the extras folder inside the package cmdbuild.zip available on the website www.cmdbuild.org
- If any additional information are required, see the official printing informations at: http://docs.geoserver.org/maintain/en/user/extensions/printing/configuration.html

11.3. Configuring GeoServer in CMDBuild

After logging into CMDBuild and going to the Administration Module, the GIS module can be enabled under the relative page in the System Config section.

To change the configuration of the GIS Service various tabs are provided to change the geoserver layers and their order.

12. BIM

12.1. Introduction

CMDBuild offers the possibility of visualizing BIM models and connect cards and bim objects.
In previous version CMDBuild made this possible with the usage of BIMServer a service that stored and provided different APIs to visualize BIM data through ifc files.

From version 3.4 a new BIM viewer has been introduced, the opensource Xeokit.
More information in the following chapters.

12.2. BIMServer introduction

BIMServer is an opensource software made to easily build bim related tools. In order to use ifc files in CMDBuild, BIMServer has to be installed.

12.3. Installation

CMDBuild supports the version 1.5.138 of BIMServer, this can be obtained on the official GitHub of BIMServer at the following link: https://github.com/opensourceBIM/BIMserver/releases .

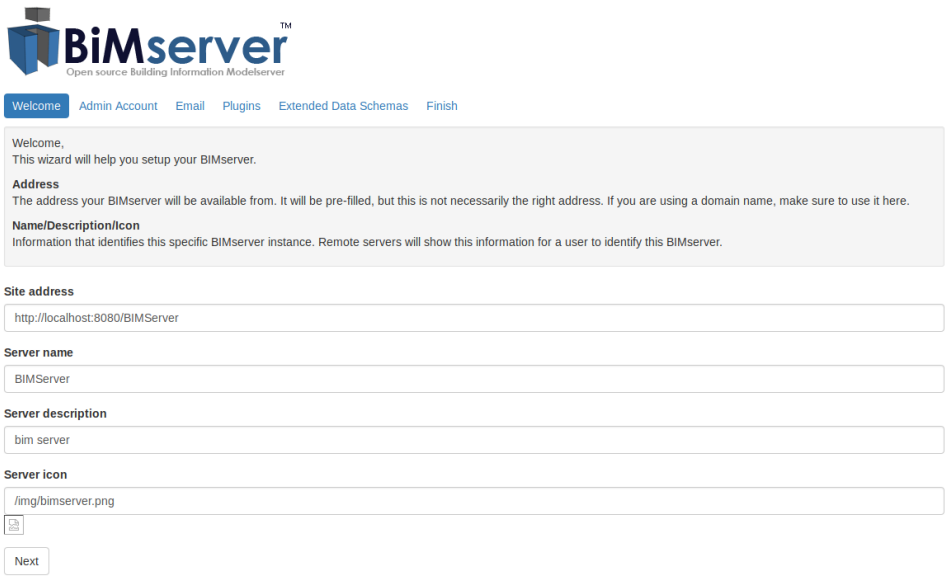
After obtaining the file we can start with the installation procedure:

BIM server is a web application, you can create a new folder called "bimserver" in your local apache tomcat installation under the webapps folder.

Now extract the war file in the newly created folder.

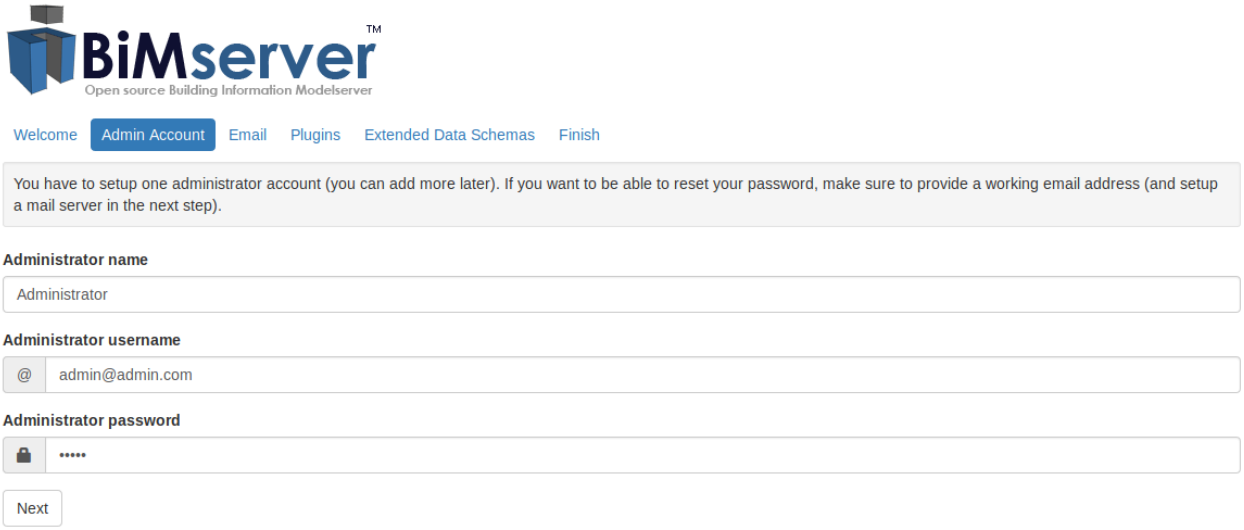
You can launch tomcat and take a look at the logs in the file "catalina.out", there the initial setup of BIMServer will be displayed.

Once the app is running proceed to the address http://localhost:8080/BIMServer/ on any browser to reach the configuration page for BIMServer:

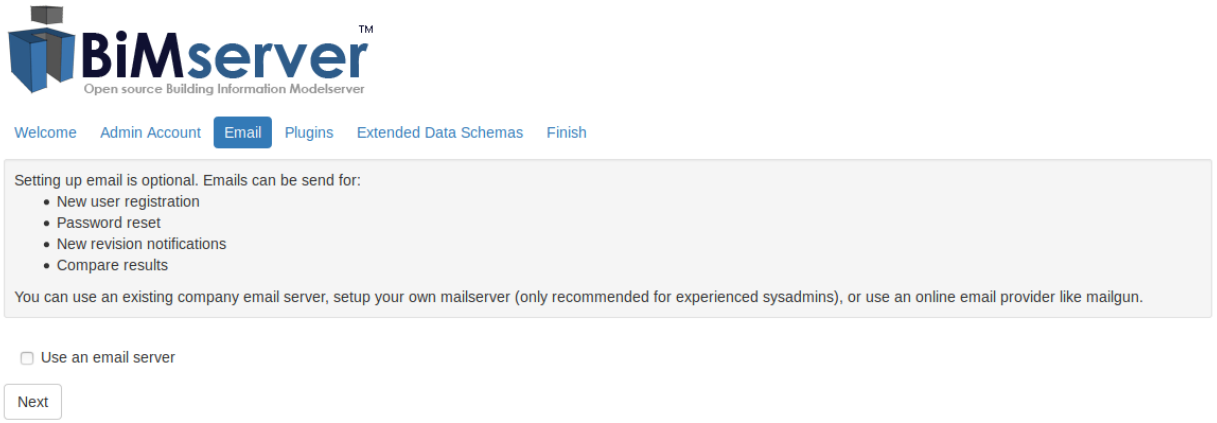


Leave the Site address as it is and choose your server name and description.

After pressing next we'll be required to decide the administrator username and password, type your preferred once, the username has to be an email address.



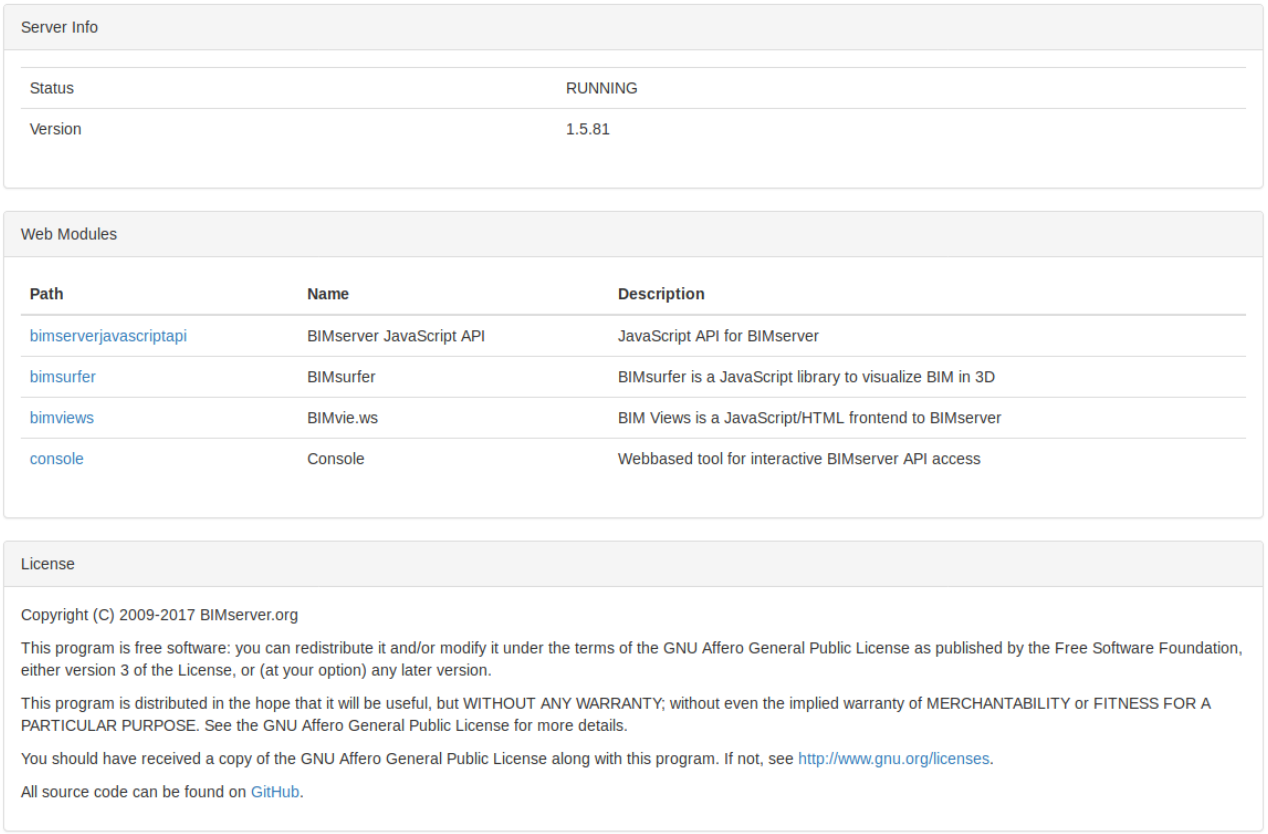
In the next step we have the possibility of setting up an email, step that isn't mandatory and won't be done here



In the next page a list of standard plugins will be provided, this plugins will be installed for an optimal setup of BIMServer, click next.

In the last page by pressing "Setup" the full installation and setup of BIMServer will start, both the page and catalina.out log will tell us the stage of the installation and when it is completed.

Once the setup is finished you can refresh the page to obtain a status page of your BIMServer:



All you need to do to complete the setup of the BIM service is to go in the administration module of CMDBuild and enable the BIM services with the information you provided during the BIMServer setup.

12.4. Xeokit introduction

Xeokit is an open source 3D graphics SDK for BIM and AEC. It utilizes a binary model format that compared to the ifc counterpart is extreamly lightweight (speaking of file size the ratio is around 10:1).

The xeokit viewer has been introduced inside the CMDBuild code as an alternative to BIMServer. A procedure to migrate ifc files to Xeokit xkt files has also been included.

If there is the need to migrate bim projects to the new file format in the bim project list in administration a button will perform the automatic conversion from ifc to xkt, and will keep stored on the database both the ifc and xkt file.

Once all files have been migrated, bimserver can be decommissioned.

When the xeokit viewer is in use, users won't see any differences in the usage because they will still upload the ifc file, CMDBuild internally will convert the ifc file to xkt and keep both versions inside the database.

General configs:

Config	Type	Description
org.cmdbuild.bim.enabled	Boolean	Bim service enable flag
org.cmdbuild.bim.viewer	String	Bim service viewer (bimserver or xeokit)
org.cmdbuild.bim.conversiontimeout	Integer	Conversion timeout for ifc→xkt file conversion

Bimserver related configs:

Config	Type	Description
org.cmdbuild.bim.bimserver.enabled	Boolean	Bim server enable flag
org.cmdbuild.bim.bimserver.url	String	Bim server URL
org.cmdbuild.bim.bimserver.username	String	Bim server username
org.cmdbuild.bim.bimserver.password	String	Bim server password

13. CMDBuild configuration in cluster mode

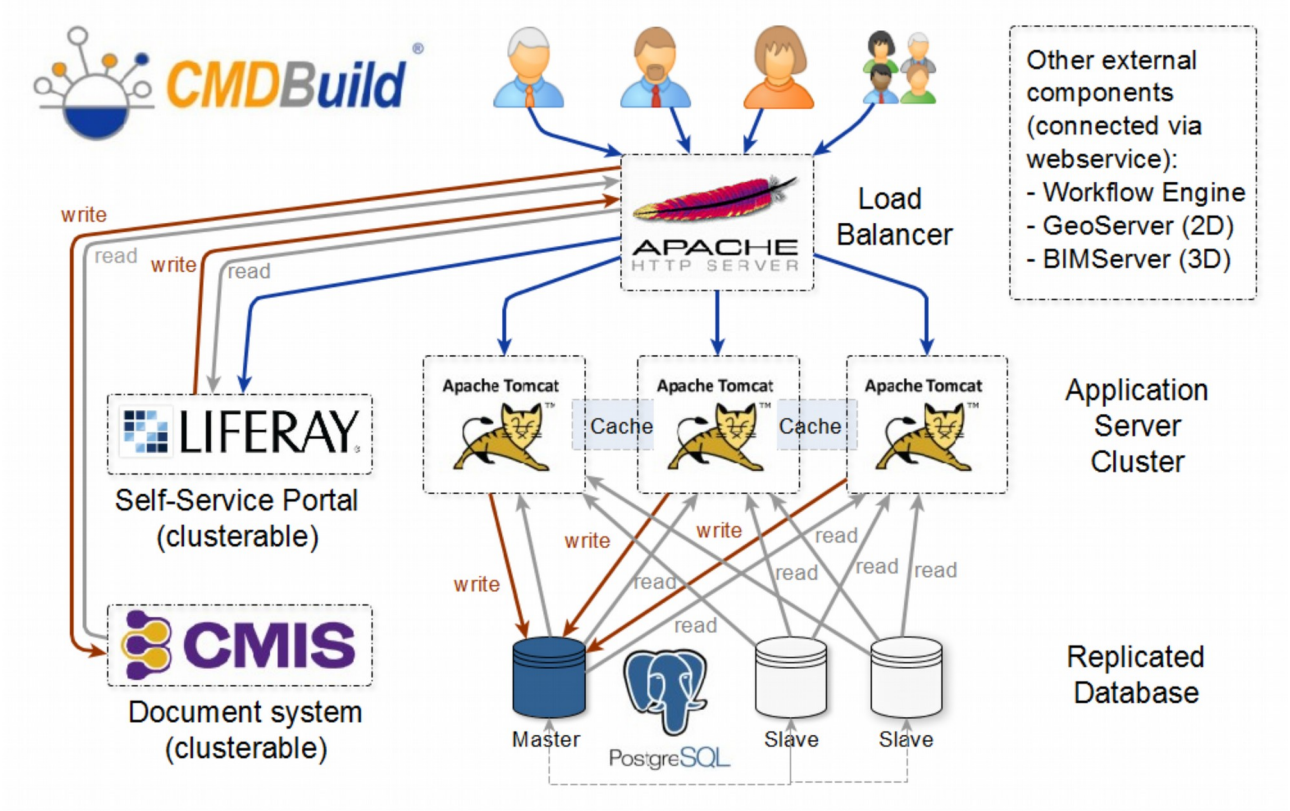
13.1. Cluster introduction

CMDBuild allows you to configure an architecture in Cluster mode, where a webserver load balancer is activated. This balancer distributes the requests to CMDBuild on various instances of the application (nodes), connected through the same database PostgreSQL, which is configured in replication mode Master-Slave.

The service clustering allows you to achieve to goals:

- High Availability: if a CMDBuild instance stops working, other instances can keep the service on
- Scalability: the availability of various instances (nodes) allows to share the charge and so to meet quicklier the requests.

The following schema describes the CMDBuild architecture in Cluster configuration:



In the example above the load balancer has been made by the Apache webserver.

Both load balacer and PostgreSQL database can be redundant through different techniques, which are not considered in this chapter.

13.2. Cluster configuration

In this chapter we will simulate the creation of a CMDBuild architecture in Cluster mode, composed by four servers:

- One server with PostgreSQL as the master database
- One server with an instance of CMDBuild (Node 1) with address 10.0.0.100:8080
- One server with an instance of CMDBuild (Node 2) with address 10.0.0.101:8080
- One server with Apache as the Load Balancer

Below, the path of the webapp cmdbuild within the servers will be identified with *${cmdbuild_home}*.

The folder including the configuration files *${cmdbuild_home}/WEB-INF/conf* has to be synchronized between the various Tomcat Instances: in case the Tomcat instances are located in different servers, you are suggested to load the folder ***conf*** onto the various instances through NFS, so that you do not have to manage the alignment of several configuration files.

Assuming that CMDBuild has been installed on the two servers by following the installation guide we can now proceed with the following steps:

Step 1:

The configuration of cmdbuild of both servers has to be changed to tell the application to enable the cluster mode.

The clustered variable has to be set to true:

```
org.cmdbuild.cluster.enabled true
```

This can be performed via rest command, by editing the variable with a setconfig command or by visualizing the entire configuration with the command editconfig.

Step 2:

Now the service which mantains the communication among the various instances (JGroups) has to be configured.

The configuration of the communication among the instances described in this manual is based on a stack of standard protocols. TCP is the transmission protocol it uses.

The configuration that is described below allows the discovery among instances which can be reached through an IP address or domain name, and should cover the majority of installation types.

If the objective is to start multiple nodes on the same host than the configuration for the tcp port has to be changed on one of the nodes with the following configuration:

```
org.cmdbuild.cluster.node.tcp.port
```

Now the list of the nodes has to be set via:

```
org.cmdbuild.cluster.nodes = 10.0.0.101,10.0.0.102
```

13.3. Configuring the load balancer on Apache

Before configuring the load balancer make sure that all the following Apache modules are functional:

- proxy
- proxy_http
- proxy_balancer
- lbmethod_byrequests
- headers

Then the configuration file of the load balancer module has to be modified.

It is also necessary to configure the desired VirtualHost by activating it in the sites-enabled to setup a domain to access the cluster.

13.4. Check if the Cluster is working

In order to verify that the clustering is working as intended you can control the table content:

quarz_scheduler_state

which has to be present in the schema *quarz* in the CMDBuild database.

If everything is working properly for every configured Tomcat node (in our case Node1 and Node2) a record will appear in the table and the value of the column *last_checkin_time* will be updated in a few seconds.

13.5. Applying patches in Cluster mode

If CMDBuild is configured in Cluster mode and the web application has to be updated, the following procedure has to be followed:

- After shutting down CMDBuild update every single CMDBuild webapp
- Start a single instance
- Reach the started instance via browser and apply any sql patches
- Start the other instances of the cluster

14. Appendix: Glossary

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.

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

See also: Card

ACTIVITY

Activity: workflow step.

An activity can be an interaction with the operator (interactive) or a script that processes operations via API (automatic).

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

See also: Process

ATTRIBUTE

The term refers to an attribute of a CMDBuild class (for example in "supplier" class the attributes can be: name, address, phone number, etc.).

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

In the database, every attribute is related to a column in the table which implements the associated class and corresponds, in the Data Management Module, to a data entry field of the specific card for the class management.

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

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).

CMDBuild includes a connector to sync some CI information (technical or maintenance records) and an interactive viewer for the 3D model of the building represented by the IFC file.

See also: CI, GIS

CI

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

In CMDBuild, the term is applied to a generic context of Asset Management extending the concept usually used in the management of IT infrastructure.

CI examples include: server, workstation, software, plant, electric panel, fire extinguisher, furniture, etc.

See also: Configuration, ITIL

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 company, a building, an asset, a service, etc.

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

A class is represented in the database with a table automatically generated when defining the class and corresponds - in the Data Management Module - to a card for the consultation and update of the cards expected in the model.

See also: Card, Attribute

CMDB

ITIL best practice (Information Technology Infrastructure Library), which has become a "standard de facto" and a non-proprietary system for services management, has introduced the term CMDB referred to the Configuration Item database.

CMDBuild extends the concept of CMDB applying it to a generic Asset Management context.

See also: Database, ITIL

CONFIGURATION

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

Even if it known as one of the main processes within the ITIL Best Practice, the same concept is used in CMDBuild for generic contexts of Asset Management.

See also: CI, ITIL

DASHBOARD

In CMDBuild, a dashboard corresponds to an application page including one or more different graphic representations, in this way you can immediately hold in evidence some key parameters (KPI) related to management aspects of the Asset Management service.

See also: Report

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.

The Asset Management database, implemented on the basis of the CMDBuild logics and philosophy, is also indicated as CMDB (Configuration Management Data Base).

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

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, or configured by the Administrator and made available by operators.

See also: Class, View

GIS

A GIS is a system able to produce, manage and analyze 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 (transmission lines) and polygons (floors, rooms, etc.).

See also: BIM

GUI FRAMEWORK

It is a framework provided by CMDBuild to simplify the implementation of external custom user interfaces and to grant a simplified access to non-technicians. They can be issued onto any webportals and can be used with CMDBuild through the standard REST webservice.

The CMDBuild GUI Framework is based on javascript JQuery libraries.

See also: Mobile, Webservice

ITIL

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

ITIL processes include: Service Support, Change Management and the Configuration 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).

CMDBuild uses some ITIL concepts and applies them to a generic context of Asset Management.

See also: Configuration

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 (also called multiple choice or picklist).

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

See also: Attribute type

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 CMDB through the REST webservice.

See also: Webservice

PROCESS

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

For each process (type of process) a new process instance will be started when users have to carry out an action on assets according to a procedure implemented as workflow.

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

The workflows managed in CMDBuild are described in the standard markup language XPDL (XML Process Definition Language), ruled by the WFMC (WorkFlow Management Coalition).

See also: Workflow step

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 CMDB.

See also: Class, Domain

REPORT

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

The reports can be configured in the Administration Module importing in XML format the description of the layout designed with the visual editor JasperReports. They can be provided to operators in the application menu.

CMDBuild users can print reports using the Management Module, which will result as printouts, charts, documents, labels, etc.

See also: Class, Domain, Database

CARDS

The term "card" refers to an element stored in a class (corresponding to the record of a table in the database).

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

SUPERCLASS

A superclass is an abstract class used as template to define attributes shared between subclasses. From the abstract class, or from abstract class hierarchies, you can derive real classes that contain data and include both shared attributes (specified in the superclass) and specific subclass attributes, besides the relations on the superclass domains and on specific domains.

For example, you can define the superclass "Company" with some basic attributes (VAT number, Business name, Phone number, etc.) and the derived subclasses "Customers" and "Suppliers", each one with both generic attributes of the superclass and its own attributes and relations.

See also: Class, Attribute

TENANT

A "tenant", in CMDBuild, is a part of the CMDB reserved to users belonging to a suborganization of the CMDBuild instance (a Group Society, a Seat, a Division, etc.).

Working in "multitenant" mode, every user works only on data of his/her suborganization and, in case, on common data: "tenants".

The list of usable Tenants can be defined from an applicable class of CMDBuild (seats, companies, customers, etc.) or from a database custom function, where you can implement complex visibility rules.

ATTRIBUTE TYPE

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

The type of attribute and its management modes are defined in the Administration Module.

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

VIEW

A view includes cards defined with logic criteria of filters applied to one or more CMDB classes.

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

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 information and applications to interact with its methods.

CMDBuild includes a SOAP and a REST webservice, which are provided to external applications to read or write data on CMDB or process operations.

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 consult or insert data or process other operations.

CMDBuild includes some standards widgets to process the most common operations, but it also supplies the specifications to implement other custom widgets.