Version

2.4

CMDBuild®

» Connectors Manual

March 2016

Author Tecnoteca srl

www.tecnoteca.com

ENG

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

CMDBuild ® is a project of Tecnoteca Srl. Tecnoteca is responsible of software design and development, it's the official maintainer and has registered the CMDBuild logo.

In the project also the Municipality of Udine was involved as the initial customer.

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

Contents

Introduction

CMDBuild is an Open Source web application designed to model and manage assets and services controlled by the ICT Department, therefore it handles the related workflow operations, if necessary according to ITIL best practices.

The management of a Configuration Database (CMDB) means keeping up-to-date, and available to other processes, the database related to the components in use, their relations and their changes over time.

With CMDBuild, the system administrator can build and extend its own CMDB (hence the project name), modeling the CMDB according to the company needs; the administration module 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.

CMDBuild provides complete support for ITIL best practices, which have become a "standard de facto" by now, a non-proprietary system for services management with process-oriented criteria.

Thanks to the integrated workflow engine, you can create new workflow processes with external visual editors, and import / execute them inside the CMDBuild application according to the configured automatisms.

A task manager integrated in the user interface of the Administration Module is also available. It allows to manage different operations (process starts, e-mail receiving and sending, connector executions) and data controls on the CMDB (synchronous and asynchronous events). Based on their findings, it sends notifications, starts workflows and executes scripts.

CMDBuild includes also JasperReports, an open source report engine that allows you to create reports; you can design (with an external editor), import and run custom reports inside CMDBuild.

Then it is possible to define some dashboards made up of charts which immediately show the situation of some indicators in the current system (KPI).

CMDBuild integrates Alfresco, the popular open source document management system. You can attach documents, pictures and other files.

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

The system includes also a SOAP and a REST webservice, to implement interoperability solutions with SOA.

CMDBuild includes two frameworks called Basic Connector and Advanced Connector, which are able - through the SOAP webservice - to sync the information recorded in the CMDB with external data sources, for example through automatic inventory systems (such as the open source OCS Inventory) or through virtualization or monitoring systems.

Through the REST webservice, CMDBuild GUI Framework allows to issue custom webpages on external portals able to interact with the CMDB.

A user interface for mobile tools (smartphones and tablets) is also available. It is implemented as multi-platform app (iOS, Android) and linked to the CMDB through the REST webservice.

Available documentation

This manual is aimed to illustrate the configuration of connectors through which it is possible to

synchronize information, managed in external applications and databases, in CMDBuild, in the different technical modes currently available.

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

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

Interoperability solutions

Feeding the CMDB

The management of IT services by large and medium-sized institutions and companies is necessarily carried out through more specialized information systems that must be able to cooperate in the management of their activities and information.

Gathering and checking manually the information managed in the CMDB can cause delay issues or create inaccuracies when updating data. Therefore it is better to update it automatically.

Therefore the configuration of connectors through external systems becomes important in order to sync in CMDBuild (the central CMDB system) data that are mainly managed ("master") on other specialist applications, among these:

- systems of automatic inventory, for automatic comparison of the technical data of the assets and of the installed software and management of the observed differences (OCS Inventory or other products)
- control systems (virtualization, etc.) to extract information about the current configuration of the IT infrastructure
- tracking systems of information regarding the execution of server services and applications
- monitoring systems, to detect system failures and start workflows of Incident Management
- LDAP directory as a repository for the archive of the staff
- HR systems, as alternative solution to receive changes to the staff list
- ERP systems to receive administration data (sources, suppliers, etc.)

General characteristics of a connector with external systems

Generally speaking, a connector with external systems is allowed to:

- contact the system of interest (automatic inventory tool, LDAP directory, etc.), through one of the possible channels of communication (direct access to DB, webservice SOAP, REST webservice, other API, structured e-mails, csv files, etc)
- access to the specific information to be synchronized (list of assets, list of people, etc.), extracted by specific SQL query or SOAP calls or API, etc.
- apply the appropriate criteria of "mapping" between the source information and the corresponding information to be inserted in CMDBuild, considering:
    - key: source data <=> key: CMDBuild class
    - entity/ data source information (such as table/ database column or item/property returned from a SOAP method) <=> entity/target information (attribute/class of CMDBuild)
    - any application logic element to be applied within the framework of the "mapping" (e.g. application of rules of aggregation/disaggregation of information, performance of operations such as log or sending notifications, posting information on the external system, etc.)

A connector will then be limited to synchronize out-of-date data by comparison to the external "master" system, or it might submit the need for change in that information to an authorization

process of Change Management (suggested for differences that are classified as "critical").

In a same instance CMDBuild can be useful to enable multiple connectors with several external systems, each with its own processing and scheduling rules.

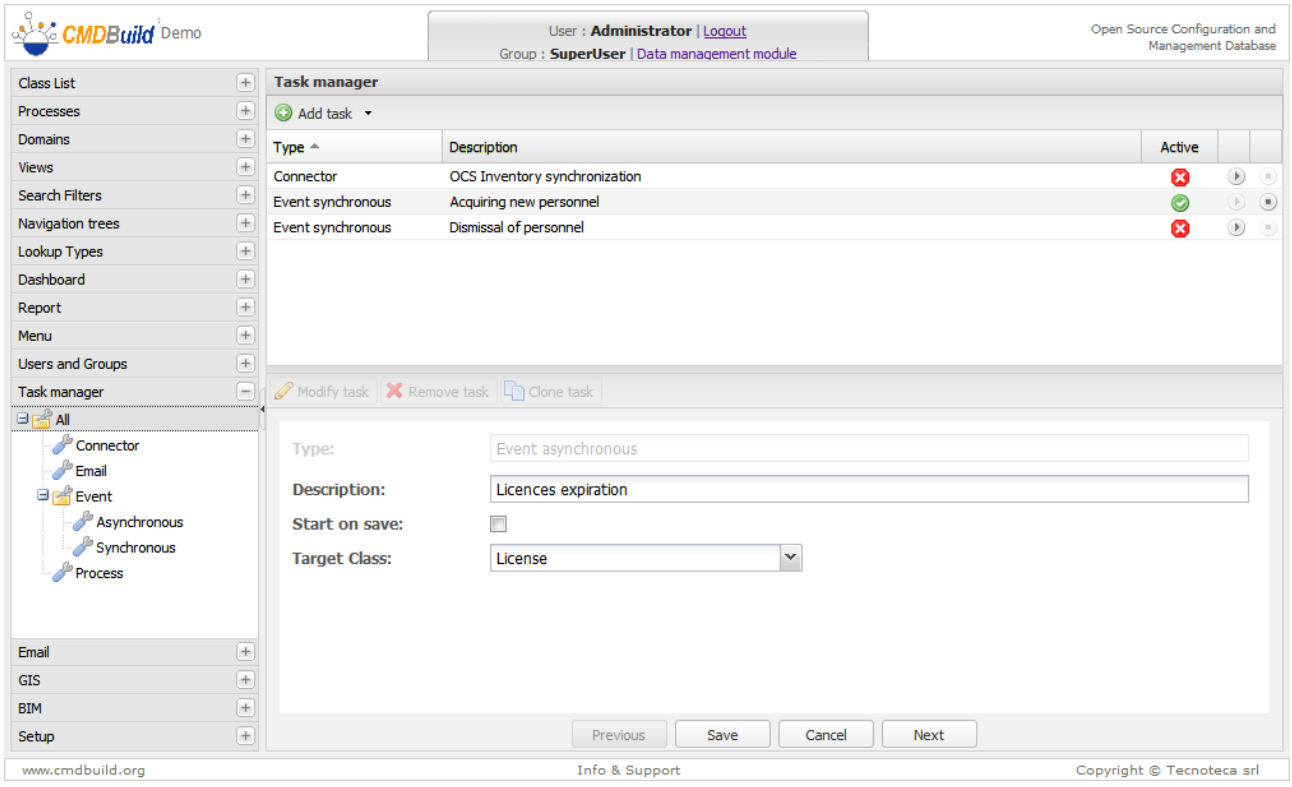
Avilable solution in the CMDBuild system

CMDBuild offers three different solutions to enable and schedule connectors to synchronize data with external systems:

- a first solution (Connector Wizard), fully configurable from the user interface through "wizard", which allows to solve the simplest cases where the rules of "mapping" do not require a specific application logic
- a second solution (Basic Connector), configurable throughthe XSLT transformation language and based on direct access to the<f1> external data source </f1> <f2></f2><f3></f3><f4>, that allows to handle some aspects of application logic (compatible with XSLT/XPATH formalism) without requiring any programming code</f4>
- a third solution (Advanced Connector), based on a framework that is programmable through the Groovy scripting language, allowing to solve even the most complex cases (different data sources, complex application logic, large amount of data)

Scheduler

The three types of connectors are all "schedulable" from the UI of CMDBuild through a special tool called "Task Manager", which is available in the Administration module.



By accessing this function the GUI of CMDBuild presents, in the upper panel, the list of configured

jobs (connectors with external systems, but also other types of activity), with the chance of:

- start a task
- suspend a task
- delete a task

Moreover, the lower panel shows the configuration parameters of the currently selected job and the icons to:

- create a new task, by specifying the configuration parameters
- modify the configurations parameters of the selected task

With every new system-startup CMDBuild will automatically restart all the jobs with "state active".

More detailed information about the different typologies of tasks and the related configuration parameters are written in the Administration Manual.

Wizard Connector

General Information

The version of the connector, based on the internal wizard, allows to resolve the simplest cases of data synchronization, in which the "mapping" rules do not require a specific application logic.

Since it is fully configurable from the user interface, it also has the advantage that it can be used immediately, without the need for complex activation procedures.

On the other hand the Wizard Connector has some limitations:

- it can handle only simple mapping rules
- it only accesses external data sources like relational database (PostgreSQL, MySQL, Oracle, SQLServer)
- each instance can synchronize only a CMDBuild class and detail top-level classes associated to the domains 1: N
- it does not comtemplate the possibility of starting authorization processes of Change Management for the resolution of the variations observed on imported data

Operation modes

The Wizard Connector is implemented as a function that is internal to the CMDBuild application and it is based on the following criteria:

- it must have access to an external data source of the expected typology
- in the external data source a subset of information of interest of CMDBuild is provided, already organised in order to be directly "mapped" on classes/attributes of CMDBuild (tables or views or other data structures for data sources that are different from relation databases)
- CMDBuild administrator, from the user interface, configures the criteria of informatio "mapping"
- a task will run at defined intervals of time during the process of scheduling:
    - it will read the information from the configured views
    - it will compare the information with the corresponding information in CMDBuild
    - it will perform updating operations
    - at the end, it will send a notification e-mail or it executes a script, according to the required configuration modalitie

Configuration modes

The Wizard Connector is one of the different tasks managed in CMDBuild with the Task Manager.

It is configured through a proper wizard requiring all the necessary information to schedule and execute a task.

For detailed information about the connector configuration you can refer to the Administration Manual.

License to use

Since this is a UI features of CMDBuild, the same open source license AGPL 3.0 of the basic application must be applied.

Basic Connector

General Information

The Basic Connector is a connector consisting of a java application, external to CMDBuild, "schedulable" both by using the appropriate system service, and through the Task Manager internal to CMDBuild.

It is based on the use of XML and in particular on the use of the XSLT 1 and XPATH languages, solution that allows you to:

- configuring the application via a simple editing of text file
- having more suppleness incustomizing, compared to to the Wizard Connector

The updating activities are engaged to a process of management that, properly configured, is able to submit the potentially critical changes to the approval of the responsible user (usually the Change Manager).

On the other hand the Basic Connector has some limitations too:

- it can operate only with external relational databases or LDAP repository (therefore it cannot communicate with external systems via webservice, or through files)
- it is able to synchronize only new insertions or modifications, not the cancellations
- each instance can synchronize only one main class and the detail classes connected with 1: N domain
- it does not offer particularly high performances

Operation modes

In particular, the connector is based on the following items:

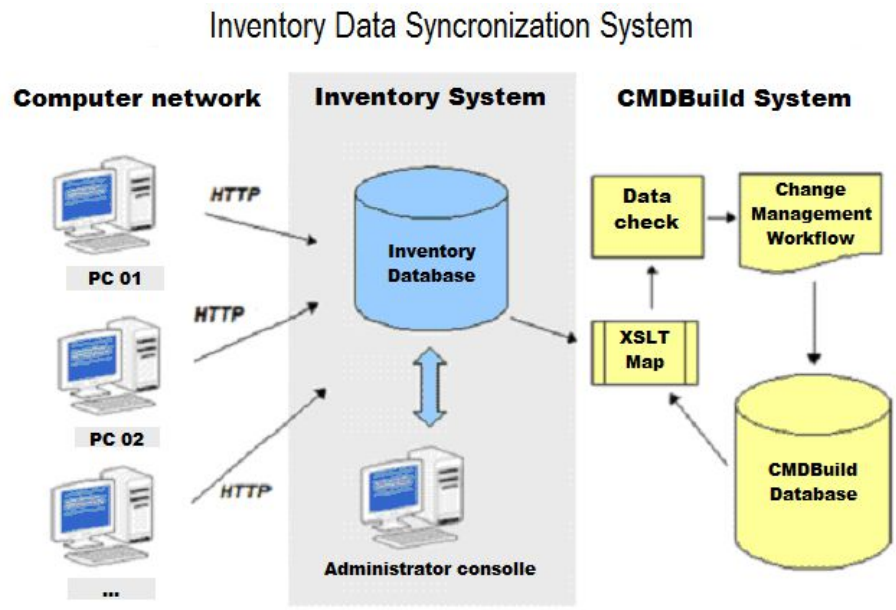
- configuration.xml: XML file that defines the access criteria to the data source (database or LDAP) and destination (CMDBuild)
- cmdbuild-schema.xml: file XML that represents the data structure of CMDBuild
- transform.xsl: file XSLT of “mapping” between the structure of the source system and that of CMDBuild, to be configured according to one's own needs for each instance
- in the case of synchronization from external databases: collection of views that should be created on the external database in order to extract the data without specifying the SQL query in the configuration files
- the java application performs the synchronization between the source and CMDbuild on the base of the configuration defined in the previous file

In particular the connector:

- queries the source by extrapolating the data of interest and transforming them according to what has been expressed in the mapping xslt file; the communication takes places via jdbc driver
- queries CMDBuild to find a matching with the data extracted from the source; the communication takes place via the webservices provided by CMDBuild
- analyzes the variations detected in the data
- for each active change it starts, via webservice, an instance of a simplified process of

"Change Management", that can be implemented according to one's own needs; in the release of CMDBuild a sample process, usable with OCS Inventory, is included. It is called "ImportAsset" and allows you to require the acknowledgment and approval by the responsible for configuration

There follows the pattern of an example instance of Basic Connector that is used to synchronize in CMDBuild all the information tracked by an application of Automatic Inventory (OCS Inventory or over):



Preliminary steps to configuring

Before syncing you need to execute some preparatory steps:

- create views from which the data will be read, just in the case of a database source
- create a process to manage changes, possibly submitting them to the review in the behalf of a figure such as the Change Manager

Views

If the external data source is a relational database the views to retrieve the extracted data must be created inside it :

- a "view" corresponding to the "catalogue" of the views illustrated in the two following paragraphs, which lists all the views on which the connector will have visibility and that will have the name of "CMDBuild_catalog"
- a "view" corresponding to the "master" table to be synchronized (eg computers noticed by the inventory tool), whose name should contain the prefix "CMDBuild _" and that should include:
    - in the first column, the unique identifier of the record in the source database, which is used to connect the main object to the linked objects
    - in the second column (please specify if coincident with the first) the identifier that will be used to unambiguously identify the object in CMDBuild (e.g. the serialnumber or the asset number or MCAddress)

    - the other attributes to follow
- multiple "views" corresponding to "slave" tabs, that are the minor entities related to the "master" view, whose name should contain the prefix "CMDBuild _" and that should include:
    - in the first column the identification value of the "master" object to which they are connected, corresponding to the first column of the "master" view, previously defined (such as the ID of the pc to which the monitor is connected)
    - the other attributes to follow

For more examples, please refer to the chapter about the connector configuration with OCS Inventory, whose description is referred to the usage of the Basic Connector.

Process of Change Management

In the CMDBuild instance a workflow of Change Management will be configured; it must be started by the Basic Connector and that must contain:

- two additional reserved attributes:
    - actionList of the TEXT type (editing: hidden)
    - actionDetail of the TEXT type (editing: editable)
- two additional assets:
    - a user activity, bootable from the group to which the user that is running the connector belongs (user specified in the configuration.xml file earlier examined)
    - a system activity that sends data to CMDBuild

If one is using the workflow engine Shark 2 it will be necessary to configure the default externalSync tool, to which the value of the variable actionList will be passed.

If one is using the workflow engine Shark 2 it will be necessary to configure the webservices in order to call the CMDBuild webservices and pass the variable actionList, by carrying out the following methods:

```
	//credential setup
	url="http://"+cm_url+"/services/soap/Private";

	//build the soap proxy for Private webservice
	soapClient = new org.cmdbuild.services.soap.client.CmdbuildSoapClient.SoapClientBuilder()
		.forClass(org.cmdbuild.services.soap.Private.class)
		.withUrl(cmdbuild_url)
		.withUsername(cmdbuild_username)
		.withPasswordType(org.cmdbuild.services.soap.client.CmdbuildSoapClient.Passw
ordType.DIGEST)
		.withPassword(cmdbuild_password)
		.build();

	proxy = soapClient.getProxy();

	proxy.sync(actionList);
```

Configuration modes

File system structure

The connector can be placed in a folder as required in the file system of the server, which from this point forward will be referred to with the name of l $ {configuration-folder-name}.

Inside that folder two subfolders named conf and logs must be created.

The three configuration files of the Basic Connector (cmdbuild-schema.xml, configuration.xml and transform.xsl) must be stored in the folder "conf", while the "logs" folder will contain the log and the log4j configuration file.

The application itself is made up of a set of libraries and an executable (connector.sh) which can be positioned at any point in the system, as long as the access to the above-mentioned files is guaranteed. The recommended position is however as follows:

Here below the suggested filesystem structure is reported.

```
|-- |--bin
|   |    |-- connector.sh
|   |
|   |--conf
|   |    |--cmdbuild-schema.xml
|   |    |--configuration.xml
|   |    |--transform.xsl
|   |
|   |--libs
|   |    |-- …
|   |
|   |-- logs
|        |--ExternalConnector.log
|        |--log4j.conf
```

The configuration.xml file

The file is divided into two sections:

The "cmdbuild" section is used to configure the connection to CMDBuild and will contain the tags:

- ServerAddress: address of the CMDBuild server (es: 127.0.0.1)
- ServerContext: CMDBuild instance name (ex: cmdbuild-test)
- ServerPort: port through which CMDBuild is reachable (e.g. 8080)
- Username: user of CMDBuild
- Password: password of the previously specified user

The "inventory"/"serverldap" section is used to configure the connection to the data source and, if the external data source is a relational database, it will contain the tags:

- DbType: the type of database to use (values supported: mysql, oracle, postgres, sqlserver, sybase)
- DbAddress: server address (ex: 127.0.0.1)
- DbPort: database port (e.g. 3306)
- DbName: database name (ex: ocsweb)
- DbUser: database user
- DbPassword: password of the previously specified user

If the external data source is an LDAP directory/Active Directory the tags will be the following:

- ServerAddress: LDAP server address (ex: 127.0.0.1)
- Port: port through which LDAP is reachable (e.g. 389)
- User: database user
- Password: password of the previously specified us

- SSL: indicate 0 if SSL is off, otherwise 1
- DNS: domain name from which to extract data (ex: ou =, dc = tecnoteca cmdbuild, dc = com)
- AdditionalDN
- Search: any filter one wants to set (ex: objectClass = user or objectClass = * to extract all objects)
- Class: name of the OU to be synchronized (i.e. personal)
- Attributes: name of the attributes to be extracted, separated by comma (e.g. cn, mail, sAMAccountName)
- Key: identification attribute of the object to be synchronized

For an example, please refer to the chapter about the configuration of the connector with OCS Inventory.

The cmdbuild-schema.xml file

The file, which describes the portion of the data structure of CMDBuild under synchronization, contains:

- the xml fileheader
- the CMDBuild tag that will contain all the tags regarding all the classes
- a tag for each class of CMDBuild (e.g. Computer and Monitor), containing the name and not the class-description
- for each class there must be at least the names (not the descriptions) of the attributes involved in the update

```
<PC>
       <Code />
       <Description />
       <SerialNumber />
       <RAM />
       <IPAddress />
       <CPUNumber />
       <CPUSpeed />
</PC>
```

For an example, please prefer to the chapter on the configuration of the connector with OCSInventory.

The transform.xls file

The file, which describes the portion of the data structure of CMDBuild under synchronization, contains:

The file, which describes the mapping of the attributes between the external data source and CMDBuild, mustalways contain the following hierarchy of xsl tags required for the transformation:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
     <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
     <xsl:template match="/">
       <CMDBUILD>
           <xsl:apply-templates/>
       </CMDBUILD>
     </xsl:template>
     <xsl:template match="*">
```

```
      <xsl:apply-templates/>
     </xsl:template>
    <xsl:template match="text()">
    </xsl:template>
   ...
  </xsl:stylesheet>
```

The tags necessary to create the actual mapping (replacing the "dots") must be inserted into that hierarchy.

In particular, there shall be two tag containers for each class that one wants to synchronize (for example "PC").

```
<xsl:template match="/inventory/device">
      <PC key="Code">
            ...
      </PC>
</xsl:template>
```

The first tag must describe the correspondence between the fields in the external data source and CMDBuild, specifying which attributes of the source must be imported and with which attributes of CMDBuild they do match. The "match" attribute must contain the name of the object that you want to consider (without the prefix "CMDBuild _" in the case of views in the DB source).

The second tag should contain the information for making the source data comparable with CMDBuild data. Then the value of the "match" tag will be modified as shown below:

```
<xsl:template match="/CMDBUILD/PC">
      <PC key="Code">
            ...
      </PC>
</xsl:template>
```

In both tags the "key" tag, that identifies the attribute of CMDBuild to be used as a unique identifier to recognize the "master card" between source (external data source) and destination (CMDBuild), must be present.

Each of the two tags contains the attributes to be updated, and the field from which it is possible to get the value to be inserted.

For example, if you want to update the "Description" attribute of the "PC" with the value of the "NAME" column in the "CMDBuild_device" view, you will write:

```
<xsl:template match="/inventory/device">
      <PC key="Code">
            <Description><xsl:value-of select="./NAME" /></Description>
      </PC>
</xsl:template>
```

The corresponding attribute in CMDBuild will be extracted as follows:

```
<xsl:template match="/CMDBUILD/PC">
      <PC key="Code">
            <Description><xsl:value-of select="./Description" /></Description>
      </PC>
</xsl:template>
```

Of course all the transformation typologies allowed by XSLT 1 can be applied.

For example, if you add all the disks detected by an inventory system such as OCS Inventory you could write:

```
<xsl:value-of select="sum(/inventory/memory/CAPACITY)" />
```

In the case of master class-related cards, as in the case of Monitor, the previous tag pair, with their attributes, must always be specified.

We must then add:

- the domain that connects objects, specified by the attribute "domain" (es. PcMonitor)
- the indication of where the "master" object is located within the domain (if "side 1" you will write "directed", if "side 2" you will write "inverted")
- the attributes of CMDBuild that identify unambiguously the object both in the source and in the destination (e.g. "Code"), separated by commas
- in the case of the CMDBuild attributes the objid standard attribute should also be added: it allows to store the id of the card of CMDBuild to perform any possible change

```
<xsl:template match="/CMDBUILD/Monitor">
      <Monitor domain="PcMonitor" domaindirection="directed" identifiers="Code">
            <xsl:attribute name="objid">
                  <xsl:value-of select="./@objid" />
             </xsl:attribute>
            <Code>
                  <xsl:value-of select="./Code" />
            </Code>
      </Monitor>
</xsl:template>
```

For an example, please refer to the chapter about the configuration of the connector with OCS Inventory.

Execution

Once the above-described structure is created, the connector can be manually started by typing the following command (unix) into the console, from the directory which contain the executable file:

```
bash connector.sh &
```

With Windows you can create a file along the lines of the provided connector.sh which starts the synchronisation by using the following syntax:

```
java -cp "${LIB_DIR}/*" -Dfile.encoding="UTF-8"
"org.cmdbuild.externalconnector.update.UpdateCMDBuild" ${configuration-folder-
name} ${processname}
```

The performance of the connector can be programmed both via the Task Manager internal to CMDBuild and through the system scheduler (the "cron" service in Unix/Linux).

For the first case, please refer to the Administrator Manual.

For the second case, it is advisable to insert into the /etc/cron.d/ folder an additional file, such as "connectors", to specify, in accordance with the cron syntax, when to execute the different connectors.

An example of scheduling is the following:

```
00 20 * * *    root    bash /usr/local/bin/connectors.sh
```

Please remember to restart the cron service once the schedule is inserted.

Limitations of use

The Basic Connector is released with the same AGPL 3.0 open source license of the basic application and, therefore, it does not have any limitation of use.

Advanced Connector

General Information

The Advanced Connector is also composed by an application external to CMDBuild, even if it is "schedulable" both through the specific system service and through the Task Manager internal to CMDBuild.

It is based on a predefined Java framework that implements the core logics, useful for synchronization activities, and requires that the specific behavior of each instance of the connector is configured through the Groovy scripting language.

Even the Advanced Connector can be configured to interact with a management process (normally a Change Management workflow in ITIL) which submits all the potentially critical changes to the approval of a responsible user (usually the Change Manager).

It is the most complete solution among those available with CMDBuild, and it guarantees:

- the ability to work with any external application with an accessible database or webservices/API or producing text files/e-mails containing the information to be synchronized
- possibility of contextual synchronization of complex data structures
- good performance with large amounts of data
- unlimited customizations, since the connector is configurable through a standard programming language

The only limitation of the Advanced Connector is that it requires programming skills for configuring each different instance.

Operation modes

The Advanced Connector consists of a series of modules that are executed in sequence according to what has been defined in the configuration file:

{CONNECTOR_HOME}/conf/connector.properties

These modules can:

- perform input/output operations by reading from the foreign data source and writing on the CDMBuild database through its webservices
- send email notification with attached report
- process the information read in input in order to normalize the data or to make them compatible with the data structure of CMDBuild

The procedures that can be carried out by each module are described in the configuration files that are defined in the folder:

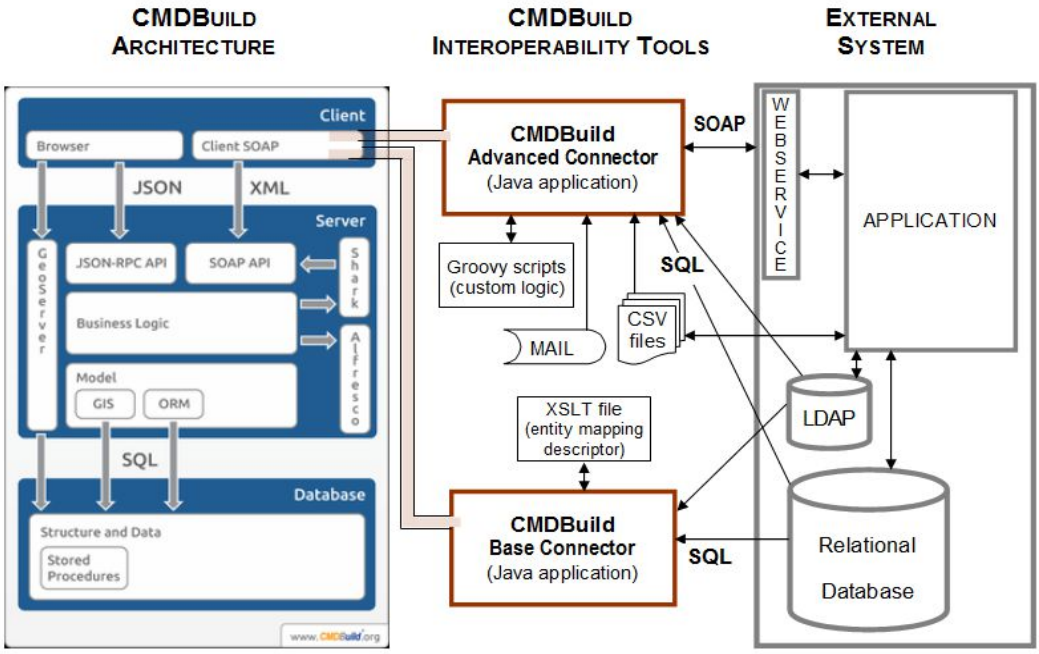
{CONNECTOR_HOME}/conf

Every configuration file consists of a script in the Groovy language that is suitably interpreted by the connector that will then perform the sequence of instructions defined in the script itself.

In detail the modules to be configured are the following:

1. module scheme: connector-schema.groovy
   It defines the scheme of the data structure of CMDBuild on which the connector keeps the data read from the external source updated.
2. sql module: connector-sql.groovy
   It is the input module that reads the information from the external data source and runs a first normalization of the read information. This module also performs a check on the read data, verifying any anomaly or inconsistency that requires the sending of a notification.
3. modify on remove module: connector-mor.groovy
   It is the output module that simply sets the connector to modify also the information that is no longer present in the external database (DBO.Data) and that would normally be removed from the CMDBuild database (in this way the information is just set as inactive and not removed from the data structure).
4. mail module: connector-mail.groovy
   It is the module that allows you to read a mailbox and to send email notifications as needed.
5. There is also an additional output module (implicit) that defines the end-point of the webservices of CMDBuild, used by the connector to write the information on CMDBuild, in the tables appropriately defined in the "schema" form.

A diagram of how the Advanced Connector and the Basic Connector interact with CMDBuild and with the external system/application with which the data synchronization is required is shown below.



Configuration modes

File system structure

The connector can be placed in a folder as required in the filesystem of the server, which from this point forward will be referred to with thename of $ {CONNECTOR_HOME}.

The default filesystem structure is shown below.

```
|-- ${CONNECTOR_HOME}
|   |
|   |--bin
|   |    |--connector.sh
|   |
|   |--config
|   |    |--connector-extension.properties
|   |    |--connector.properties
|   |    |--connector-sql-context.xml
|   |    |--log4j.properties
|   |    |--template
|   |    |    |-- ...
|   |
|   |-- lib
|        |--...
```

The bin/connector.sh file

It is the script needed to run the application. It contains the configuration and default parameters needed for the performance.

The config/connector.properties file

The file describes these modules and their organization (chains):

```
modules.schema=
modules.in.source=
modules.in.cmdbuild=ws
modules.out=
```

```
module.ws.classname=org.cmdbuild.connector.ws.WsModule
```

At the very least a scheme module and a form of input and output module must be specified. By default the necessary module for the communication with CMDBuild via web service is already present.

The directory config/template

This directory contains the templates of the configuration files for all the available modules.

The module configuration files

For each added module, within the directory ${CONNECTOR_HOME}/config, the files “connector-${MODULE_NAME}.properties” and “connector-${MODULE_NAME}.groovy”, where “${MODULE_NAME}” represents the name of the module as specified within the file.

Execution

Once the above-described files are edited, the connector can be started by running the script:

```
connector.sh
```

The execution of the connector can be programmed both via the Task Manager internal to CMDBuild and through the system scheduler (the "cron" service in Unix/Linux).

For the first case, please refer to the Administrator Manual.

For the second case, it is advisable to insert into the /etc/cron.d/ folder an additional file, such as "connectors", to specify, in accordance with the cron syntax, when to execute the different connectors.

An example of scheduling is the following:

```
00 20 * * *    root    sh ${CONNECTOR_HOME}/bin/connectors.sh
```

Please remember to restart the cron service once the schedule is inserted.

Limitations of use

The Advanced Connector is made available with non-open source license, which allows only those who have signed with Tecnoteca srl a maintenance service for the CMDBuild application to use the service, and only until that service is active.

However, the Advanced Connector is provided with the source code of all its components to those who have signed with Tecnoteca srl a maintenance service for the CMDBuild application to use the service

Examples of connectors

General Information

Being CMDBuild a totally customizable system, it is not possible to create standard connectors that work on system whose configuration is not known.

Since the aim of the connector is to synchronize information between the data model of the external system and the data model of CMDBuild, the total freedom in the configuration of CMDBuild requires to proceed with the connector configuration.

Similarly any approval mechanism of the most critical Changes allowed by the logic of the connector must be adapted to the corresponding workflows configured in CMDBuild.

For this reason the three types of connectors, described in the previous pages, contain configuration options, even if with different degrees of difficulty (UI => editing of text files => writing of programming code).

In the subsequent pages some already implemented connectors are described, that must therefore be taken as sample implementations and not as ready-to-use tools.

The configuration criteria of the connector with OCS Inventory, that is also available as a downloadable file from the site of CMDBuild, are described in more detail.

OCS Inventory

General Information

The manual control and update of the changes in the inventory of IT assets require important resources that are not always available.

It is therefore suggested to automate the synchronization of that information through the use of an automatic inventory tool and, in particular, OCS Inventory (http://www.ocsinventory-ng.org/en/) is the most suggestable product, considering both its technical features and the open source license.

OCS Inventory allows you to:

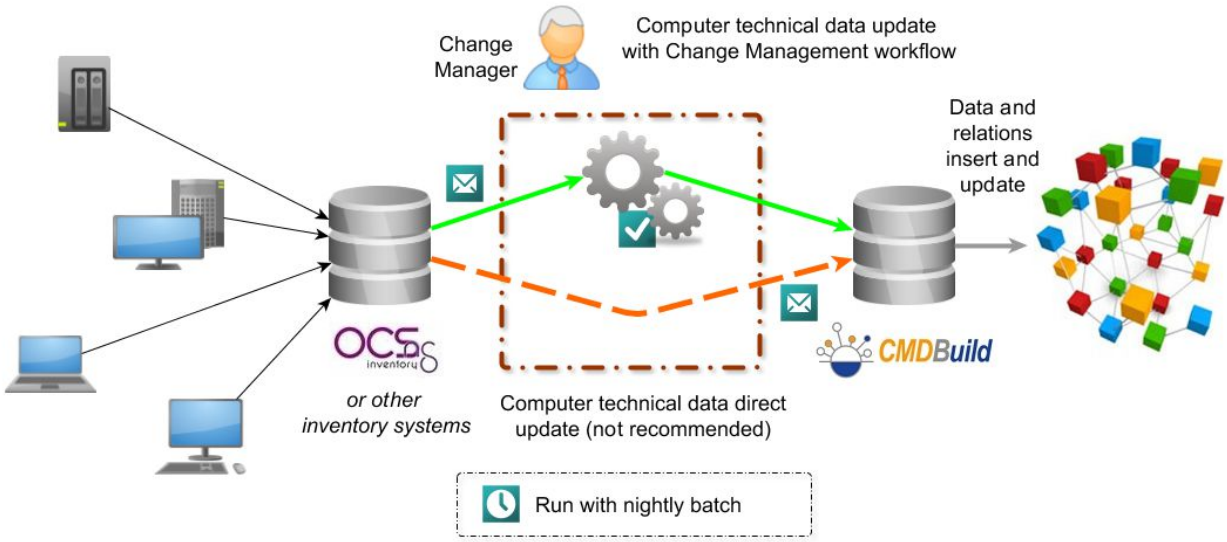
- activate "agent" programs on the computers to be controlled (Windows, Linux or Mac OS)
- collect information about BIOS, processor, RAM, input devices, controller, peripherals, additional cards, network settings, operating system, software applications, etc.
- periodically send to the server that information in the form of XML files
- store that information in a MySQL relational database
- consult and modify that information from a proper management web application

In the download page of the CMDBuild site you'll find the file:

external-connectors-x.y.z.zip (connector for import from external sources - OCS Inventory)

that implements that connector by using the above-described Basic Connector.

The explanations shown in the following pages refer to that type of connector, but similar results can be obtained thanks to the Wizard Connector too (if the logic of "mapping" is simple) or to the Advanced Connector.



Operation modes

The connector implementation is based on:

- the periodic comparison between the data stored in CMDBuild and the data collected from OCS Inventory, based on the criteria of "mapping" in the XLST transform file
- the activation of the CMDBuild workflow system with automatic change requests (simplified variant of a Change Management process) that are generated on the basis of the detected configuration changes

The second criterion does obviously depend on the explicit control and role management requirements and on the responsibilities required by ITIL.

The complete list of information made available from OCS Inventory and so mappable on the classes and attributes of CMDBuild is shown in the Appendix.

It is warmly suggested to carefully consider which attribute is to be used as computer's unique field: a number of internal asset (specify "one-off" with the appropriate flag "tag" during the OCS-Agent installation), the MAC Address of the network adapter (not necessarily invariant in case of substitution), etc.

Configuration modes

According to what has been described in the paragraphs above, an example of compilation of the three configuration files of the Basic Connector, working on the demo database released with CMDBuild, is illustrated below.

File configuration.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
	<cmdbuild>
		<ServerAddress>127.0.0.1</ServerAddress>
		<ServerContext>cmdbuild</ServerContext>
		<ServerPort>8080</ServerPort>
		<Username>inventory</Username>
		<Password>inventory</Password>
	</cmdbuild>
	<inventory>
		<DbType>mysql</DbType>
		<DbAddress>127.0.0.1</DbAddress>
		<DbPort>3307</DbPort>
		<DbName>ocsweb</DbName>
		<DbUser>root</DbUser>
		<DbPassword>root</DbPassword>
	</inventory>
</configuration>
```

File cmdbuild-schema.xml

```
<?xml version="1.0"?>
<CMDBUILD>
	<PC>
		<Code />
		<Description />
		<SerialNumber />
		<RAM />
		<IPAddress />
		<CPUNumber />
		<CPUSpeed />
	</PC>
</CMDBUILD>
```

File transform.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	version="1.0">
	<xsl:output method="xml" version="1.0" encoding="UTF-8"
		indent="yes" />
	<xsl:template match="/">
		<CMDBUILD>
			<xsl:apply-templates />
		</CMDBUILD>
	</xsl:template>
	<xsl:template match="*">
		<xsl:apply-templates />
	</xsl:template>
	<xsl:template match="text()"></xsl:template>
	<xsl:template match="/inventory/device">
		<PC key="Code">
			<Code>
				<xsl:value-of select="./TAG" />
			</Code>
			<Description>
				<xsl:value-of select="./NAME" />
			</Description>
			<IPAddress>
				<xsl:value-of select="./IPADDR" />
```

```
                </IPAddress>
                <RAM>
                        <xsl:value-of select="sum(/inventory/memory/CAPACITY)" />
                </RAM>
                <CPUNumber>
                        <xsl:value-of select="./PROCESSORN" />
                </CPUNumber>
                <CPUSpeed>
                        <xsl:value-of select="./PROCESSORS" />
                </CPUSpeed>
        </PC>
</xsl:template>
<xsl:template match="/CMDBUILD/PC">
        <PC key="Code">
                <xsl:attribute name="objid">
                <xsl:value-of select="./@objid" />
                 </xsl:attribute>
                <Code>
                        <xsl:value-of select="./Code" />
                </Code>
                <Description>
                        <xsl:value-of select="./Description" />
                </Description>
                <IPAddress>
                        <xsl:value-of select="./SO" />
                </IPAddress>
                <RAM>
                        <xsl:value-of select="./RAM" />
                </RAM>
                <CPUNumber>
                        <xsl:value-of select="./CPUNumber" />
                </CPUNumber>
                <CPUSpeed>
                        <xsl:value-of select="./CPUSpeed" />
                </CPUSpeed>
        </PC>
</xsl:template>
</xsl:stylesheet>
```

Definition of the views

As indicated in the description of the Basic Connector, there should be defined:

- a "view" corresponding to the "catalogue" of views described below
- a "view" corresponding to the "master table"
- multiple "views" corresponding to "slave" tables

View catalogues

```
CREATE OR REPLACE VIEW CMDBuild_catalog
AS select TABLE_NAME from INFORMATION_SCHEMA.VIEWS
where table_name like '%CMDBuild_%' and table_name <> 'CMDBuild_catalog';
```

'Master' view

```
CREATE OR REPLACE VIEW CMDBuild_device
      (DEVICE, TAG, DEVICEID, NAME, WORKGROUP, USERDOMAIN, OSNAME, OSVERSION,
      OSCOMMENTS, PROCESSORT, PROCESSORS, PROCESSORN, MEMORY, SWAP, IPADDR,
WINCOMPANY, WINOWNER, WINPRODID, WINPRODKEY)
as select ID, TAG, DEVICEID, NAME, WORKGROUP, USERDOMAIN, OSNAME, OSVERSION,
      OSCOMMENTS, PROCESSORT, PROCESSORS, PROCESSORN, MEMORY, SWAP, IPADDR,
WINCOMPANY, WINOWNER, WINPRODID, WINPRODKEY
from hardware, accountinfo
WHERE ID=HARDWARE_ID;
```

'Slave' views

```
CREATE OR REPLACE VIEW CMDBuild_controller
as select HARDWARE_ID, MANUFACTURER, NAME, CAPTION, DESCRIPTION, VERSION, TYPE
from controllers;

CREATE OR REPLACE VIEW CMDBuild_drive as
select HARDWARE_ID, LETTER, TYPE, FILESYSTEM, TOTAL, FREE, NUMFILES, VOLUMN
from drives;

CREATE OR REPLACE VIEW CMDBuild_input
as select HARDWARE_ID, TYPE, MANUFACTURER, CAPTION, DESCRIPTION,
      INTERFACE, POINTTYPE
from inputs;

CREATE OR REPLACE VIEW CMDBuild_memory
as select HARDWARE_ID, CAPTION, DESCRIPTION, CAPACITY, PURPOSE,
      TYPE, SPEED, NUMSLOTS
from memories;

CREATE OR REPLACE VIEW CMDBuild_bios
as select HARDWARE_ID, SMANUFACTURER, SMODEL, SSN, TYPE,
      BMANUFACTURER, BVERSION, BDATE
from bios;

CREATE OR REPLACE VIEW CMDBuild_monitor
as select HARDWARE_ID, ID, MANUFACTURER, CAPTION, DESCRIPTION, TYPE, SERIAL
from monitors;

CREATE OR REPLACE VIEW CMDBuild_modem
as select HARDWARE_ID, ID, NAME, MODEL, DESCRIPTION, TYPE
from modems;
```

```
CREATE OR REPLACE VIEW CMDBuild_port
as select HARDWARE_ID, ID, TYPE, NAME, CAPTION DESCRIPTION
from ports;
CREATE OR REPLACE VIEW CMDBuild_slot
as select HARDWARE_ID, ID, NAME, DESCRIPTION, DESIGNATION,
      PURPOSE, STATUS, PSHARE
from slots;

CREATE OR REPLACE VIEW CMDBuild_software
as select HARDWARE_ID, ID, PUBLISHER, NAME, VERSION, FOLDER, COMMENTS,
      FILENAME, FILESIZE, SOURCE
from softwares;

CREATE OR REPLACE VIEW CMDBuild_sound
as select HARDWARE_ID, ID, MANUFACTURER, NAME, DESCRIPTION
from sounds;

CREATE OR REPLACE VIEW CMDBuild_storage
as select HARDWARE_ID, ID, MANUFACTURER, MODEL, DESCRIPTION, TYPE, DISKSIZE
from storages;

CREATE OR REPLACE VIEW CMDBuild_video
as select HARDWARE_ID, ID, NAME, CHIPSET, MEMORY, RESOLUTION
from videos;
```

Available attributes and examples of "mapping"

The list of the entities and information gathered from OCS Inventory, among which those that will be "mapped" in CMDBuild, and possibly on which classes and attributes (last column), must be identified, is shown below.

OCS Inventory	CMDBuild I/F view		Description	Example	CMDBuild
Hardware.Id	CMDBuild_device	Id	Id	2	
AccountInfo.Tag		Tag	Key	lisa	
--		Serial	Serial number	HUB5390NQK	
Hardware.Name		Name	Activity name	Kelly	
Hardware.Workgroup		Workgroup	Working Group	Tecnoteca	
Hardware.UserDomain		UserDomain	Domain	Kelly	
--		Manufacturer	Producer	Hewlett-Packard	
--		ProductName	Product name	HP Compaq nc6120 (PN936AV)	
Hardware.OSName		OSName	Operating system name	Microsoft Windows XP Professional	
Hardware.OSVersion		OSVersion	Operating system version	5.1.2600	
Hardware.OSComments		OSComments	Service Pack		
Hardware.ProcessorT		ProcessorS	Processor speed	1832	
Hardware.ProcessorS		ProcessorN	Number of processors	1	
Hardware.ProcessorN		Memory	RAM Memory	512	
Hardware.Memory		WinProdKey	Product Keyo		
Hardware.WinProdKey					
Bios.Hardware_Id	CMDBuild_bios	Device_Id	Reference machine	2	
Bios.SManufacturer		SManufacturer	MotherBoard Manufacturer	VIA	
Bios.SModel		SModel	MotherBoard Model	K7Upgrade-600	
Bios.SSN		SSN	Serial number M.B.	SYS-1234567890	
Bios.Type		Type	Chassis type	Desktop	
Bios.BManufacturer		BManufacturer	Producer	American Megatrends Inc.	
Bios.BVersion		BVersion		AMIINT-10-SMBiosVersion: P1.50	

Bios.BDate		BDate	Version Signing date	N/A	
Controllers.Hardware_Id Controllers.Manufacturer Controllers.Name Controllers.Caption Controllers.Description Controllers.Version Controllers.Type	CMDBuild_controller	Device_Id Manufacturer Name Caption Description Version Type	Reference machine Producer Activity name Title Description Version Type	2 VIA Technologies, Inc. Controller IDE VIA Bus Master Controller IDE VIA Bus Master Controller IDE VIA Bus Master N/A IDE Controller	
Drives.Hardware_Id Drives.Letter Drives.Type Drives.FileSystem Drives.Total Drives.Free Drives.NumFiles Drives.VolumN	CMDBuild_drive	Device_Id Letter Type FileSystem Total Free NumFiles VolumN	Reference machine Letter Type File System Total space Free space Number of files Volume name	2 C:/ Hard Drive NTFS 39997 15708 0	
Inputs.Hardware_Id Inputs.Type Inputs.Manufacturer Inputs.Caption Inputs.Description Inputs.Interface Inputs.PointType	CMDBuild_input	Device_Id Type Manufacturer Caption Description Interface PointType	Reference machine Type Producer Title Description Interface Pointer type	2 Pointing (Standard System Devices USB Human Interface Device USB Human Interface Device USB N/A	
Memories.Hardware_Id Memories.Caption Memories.Description Memories.Capacity Memories.Purpose Memories.Type Memories.Speed Memories.NumSlots	CMDBuild_memory	Device_Id Caption Description Capacity Purpose Type Speed NumSlots	Reference machine Title Description Dimension Purpose Type Speed Number of slots	2 Physical memory DIMM1 (Other ECC) 512 Reserved Unknown N/A 1	
Modems.Hardware_Id Modems.Name Modems.Model Modems.Description Modems.Type	CMDBuild_modem	Device_Id Name Model Description Type	Reference machine Activity name Model Description Type		
Monitors.Hardware_Id Monitors.Manufacturer Monitors.Caption Monitors.Description Monitors.Type Monitors.Serial	CMDBuild_monitor	Device_Id Manufacturer Caption Description Type Serial	Reference machine Producer Title Description Type Serial number	2 Unknown manufacturer code ACR AL1716 RGB color	
Networks.Hardware_Id Networks.Description Networks.Type Networks.TypeMIB Networks.Speed Networks.MCAddr Networks.Status Networks.IPAddress Networks.IPMask Networks.IPGateway	CMDBuild_network	Device_Id Description Type TypeMIB Speed MACAddr Status IPAddress IPMask IPGateway	Reference machine Description Type MIBType Speed Macaddress State IP Address IP Mask IP Gateway	2 Fast Ethernet VIA compat. card Ethernet EthernetCsmacd 100 Mb/s 00:0B:6A:AE:DE:6B Up 192.168.2.199 255.255.255.0 192.168.2.1	

-- Networks.IPSubnet Networks.IPDHCP		IPDNS IPSubnet IPDHCP	IP DNS IP subnet IP DHCP server	192.168.2.100 192.168.2.0 255.255.255.255	
Ports.Hardware_Id Ports.Type Ports.Name Ports.Caption Ports.Description	CMDBuild_port	Device_Id Type Name Caption Description	Reference machine Type Activity name Title Description	2 Serial Communications port (COM1) Communications port (COM1) Communication port	
Slots.Hardware_Id Slots.Name Slots.Description Slots.Designation Slots.Purpose Slots.Status Slots.PShare	CMDBuild_slot	Device_Id Name Description Designation Purpose Status PShare	Reference machine Activity name Description Code Use State Sharing	2 System Slot System Slot PCI1 OK 1	
Softwares.Hardware_Id Softwares.Publisher Softwares.Name Softwares.Version Softwares.Folder Softwares.Comments Softwares.FileName Softwares.FileSize Softwares.Source	CMDBuild_software	Device_Id Publisher Name Version Folder Comments FileName FileSize Source	Reference machine Producer Activity name Version Folder Comment File name File size Source	2 OpenOffice.org OpenOffice.org 2.0 2.0.9044 OpenOffice.org 2.0 (en-US)(Build:9044) N/A 0 1	
Sounds.Hardware_Id Sounds.Manufacturer Sounds.Name Sounds.Description	CMDBuild_sound	Device_Id Manufacturer Name Description	Reference machine Producer Activity name Description	2 Silicon Integrated Systems [SiS] AC'97 Multimedia audio controller rev a0	
Storages.Hardware_Id Storages.Manufacturer Storages.Name Storages.Model Storages.Description Storages.Type Storages.DiskSize	CMDBuild_storage	Device_Id Manufacturer Name Model Description Type DiskSize	Reference machine Producer Activity name Model Description Type Disk size	2 (standard disk unity) Generic Flash Disk USB Device //./PHYSICALDRIVE1 Disk unity Removable media other thanxfloppy 117	
Videos.Hardware_Id Videos.Name Videos.Chipset Videos.Memory Videos.Resolution --	CMDBuild_video	Device_Id Name Chipset Memory Resolution RefreshRate	Reference machine Activity name Chipset Memory Resolution Refresh frequency	2 RADEON 9200 PRO Family (Microsoft C.) RADEON 9200 PRO AGP (0x5960) 128 1280 x 1024 64	

Active Directory

General Information

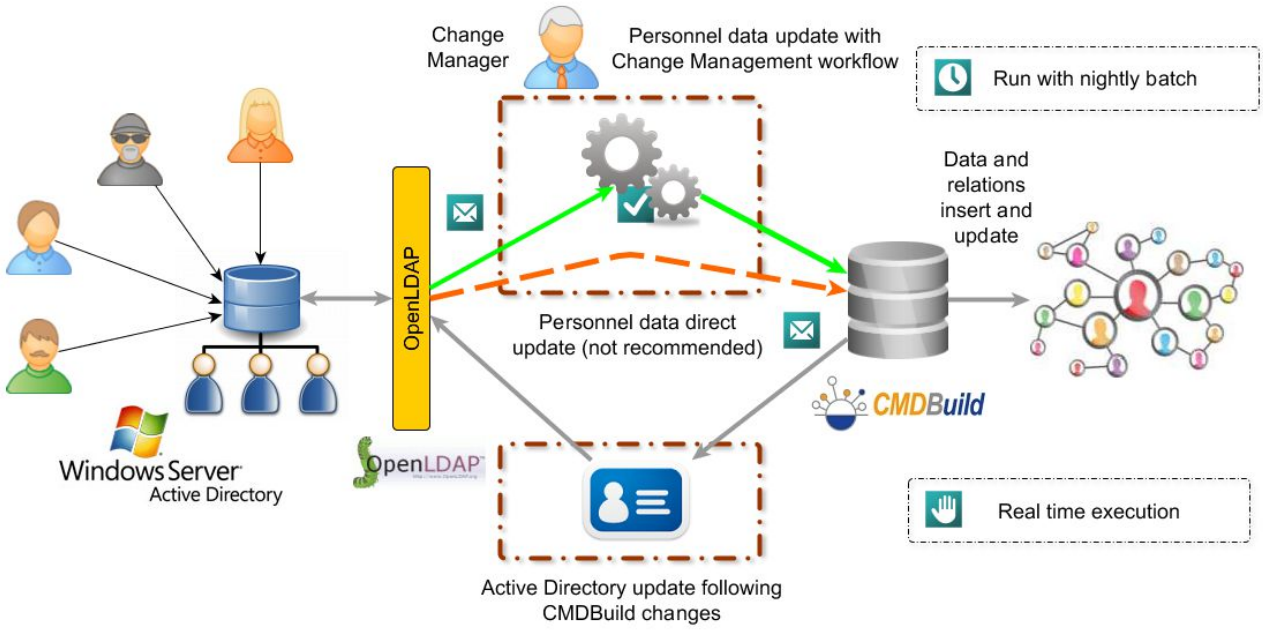
Despite being a typology of information normally treated by dedicated systems, it is generally useful to have the staff stored in CMDBuild too, thus enabling to correlate, for example, computers to their assignees, activities of assistance to applicants, rooms to their occupants, etc.

It is therefore useful to activate a connector for the automatic synchronization of data between the master archive, which is very often managed through the Active Directory service, and CMDBuild.

If the logic of "mapping" is simple, the synchronization can be achieved thanks to the Wizard Connector, otherwise thanks to the Basic Connector or to the Advanced Connector.

By using the Advanced Connector a hybrid solution is also possible, in which the main data is processed in Active Directory and synchronized in CMDBuild, while some more specific technical information (e.g. the email address) is treated as "master" in CMDBuild and updated on Active Directory (see diagram below).

Moreover, the Active Directory Connector can be connected to a Change Management workflow, useful, for example, to start running some technical operations at the entrance of new members of the staff or at the exit of those already-present (account creation/deletion, assigning/retirement of computing equipment, etc).



Operation modes

If one decides to use the Basic Connector, the configuration is obtainable from the general description of the connector and from the sample files shown in the paragraph regarding the synchronization with OCS Inventory.

In the case you have to sync several cards or you need to implement complex management logics,

the use of the Advanced Connector is suggested. It grants better performances and a better application flexibility.

Nagios – GroundWork – NetEye

General Information

Among the management tools adopted for the control of information systems, a forefront role is played by the solutions for equipment monitoring and management of alarms (we would like to remember, among the open source applications, Nagios, NetEye, GroundWork, etc).

Often, however, the use of these tools does not include input integrations into the company CMDB and the output is limited to only sending notification emails.

As a hint for those who have to configure this kind of connectors, here is shown the description of two actual uses.

Optimize the activities of the Service Desk

The aim of this first solution is to assist the operators in assessing any malfunction and starting, at the same time, the Incident Management processes to solve it.

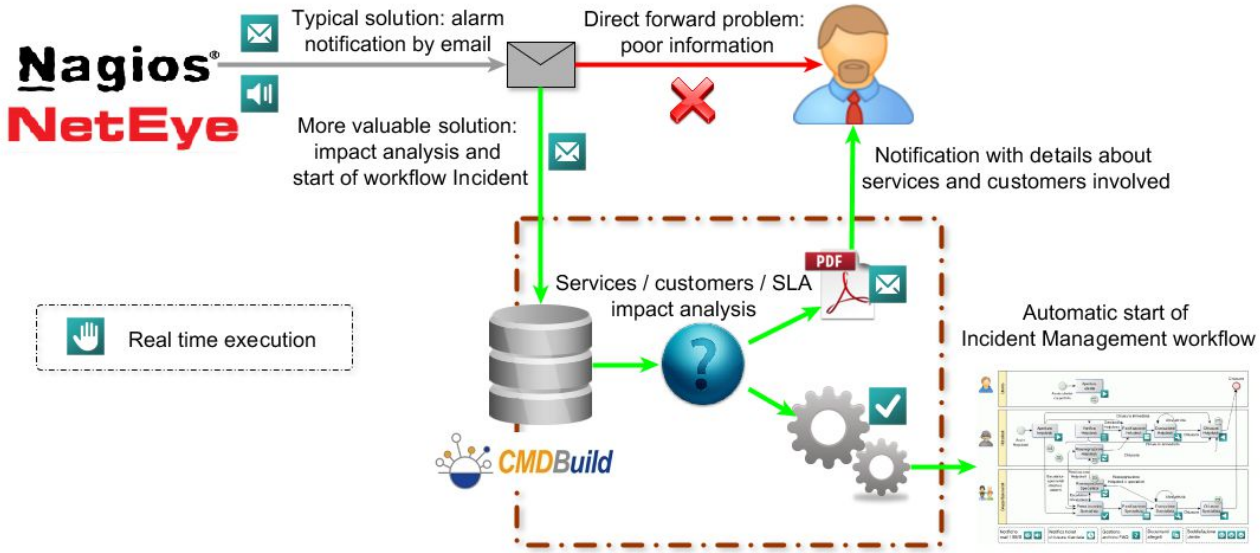
The most common solution provides that the monitoring system will send e-mails such as "Host xy down". In many cases, however, the technician who receives the notification is not able to instantly understand the seriousness of the problem and, for example, if he is outside the company headquarters on the days of availability, he does not have immediate access to the necessary additional information for a deeper and further evaluation.

The connector that is suggested to solve this kind of problems behaves as follows:

- it intercepts the e-mails sent by the monitoring system to the mailboxes (IMAP) of the technical support
- it analyzes the text of the e-mail, which, being produced by the monitoring tool, is normally based on a predefined and configurable pattern, and extracts the relevant information (for example, the seriousness of the problem and the hostname of the server)
- it uses the webservice of CMDBuild to access the CMDB and retrieve the information of interest, that, depending on the completeness of the configured data model, may include:
    - additional technical information on the server involved (memory, operating system, etc.)
    - list of the technical services activated on that server
    - list of the user services involved
    - list of the customers who have subscribed them
    - list of the SLA provided for each customer on a given service
    - critical details
- it produces a report containing the above-listed information and sends it to the technical operator, as attached file to the original e-mail reporting the problem
- it starts an Incident Management workflow in CMDBuild, precompiling the available information
- it records the event of down in a special CMDBuild entity (typically configured as "simple" class), connecting it to the server involved
- it reports a "down" in progress on the server tab involved

The configuration of a connector of this type, with an extensive application logic, requires the use of the Advanced Connector.

A schematic diagram of the solution is shown below:



For further information:

The implementation is described in a case history we have presented at the "CMDBuild Day 2012", held in Bologna, at the Emilia Romagna Regional Council last May 10th 2012.

Click on this link to watch video and slides (in Italian) of the intervention “Monitoraggio dei servizi e gestione degli allarmi con CMDBuild” (Monitoring of services and management of alarms with CMDBuild)

http://www.cmdbuild.org/it/diffusione/cmdbuild-day/cmdbuild-day-2014/2012

Optimize the configuration of the monitoring system

The setup activities of a monitoring system are far from being banal, since they have to produce the full list of hosts to be monitored, the services to be monitored on each of them and the dependencies in relation to other hosts on the network (to prevent a breakdown of a device producing hundreds of alarms sent by every other host that depends on it).

Very often such activities are manually carried out by duplicating information that should already be present in the CMDB, and forcing a double update activity in the case of changes to the network configuration (resulting in a high probability of error).

The solution, suggested by an external contributor (see references below), currently in beta version, is to enrich the data model of CMDBuild with additional elements that allow to obtain the automatic export of the configuration files of:

- Nagios / NetEye
- NagVis, plugin that allows to view the status of the checks within maps and diagrams
- Nagios Business Process, plugin that implements two main views:
    - Business View: it aggregates the result of the Nagios checks to view the status of high-level services

    - Business Impact Analysis: it allows to perform impact analysis by simulating the down of one or more components and by displaying the effects on high level services

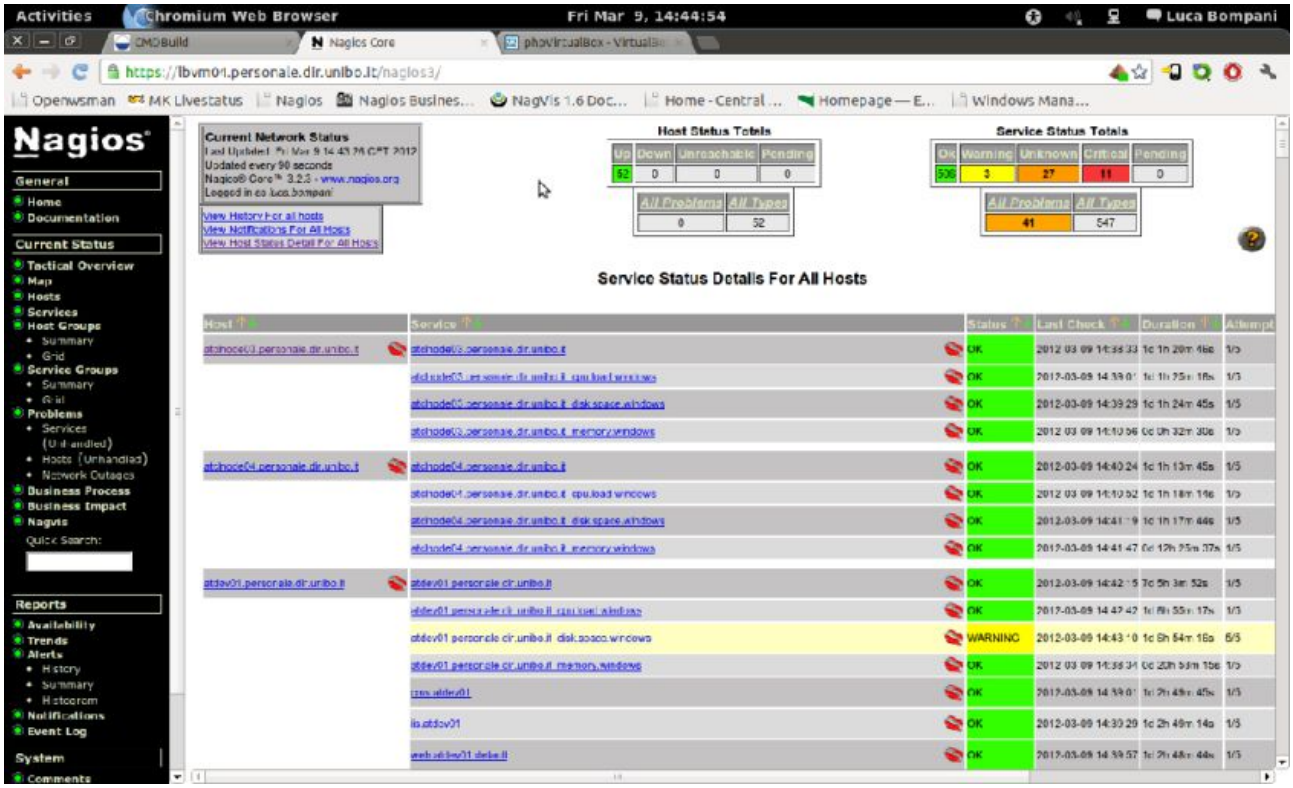
The connector consists of a Java application, different from the Basic Connector and from the Advanced Connector described above, which operates as follows:

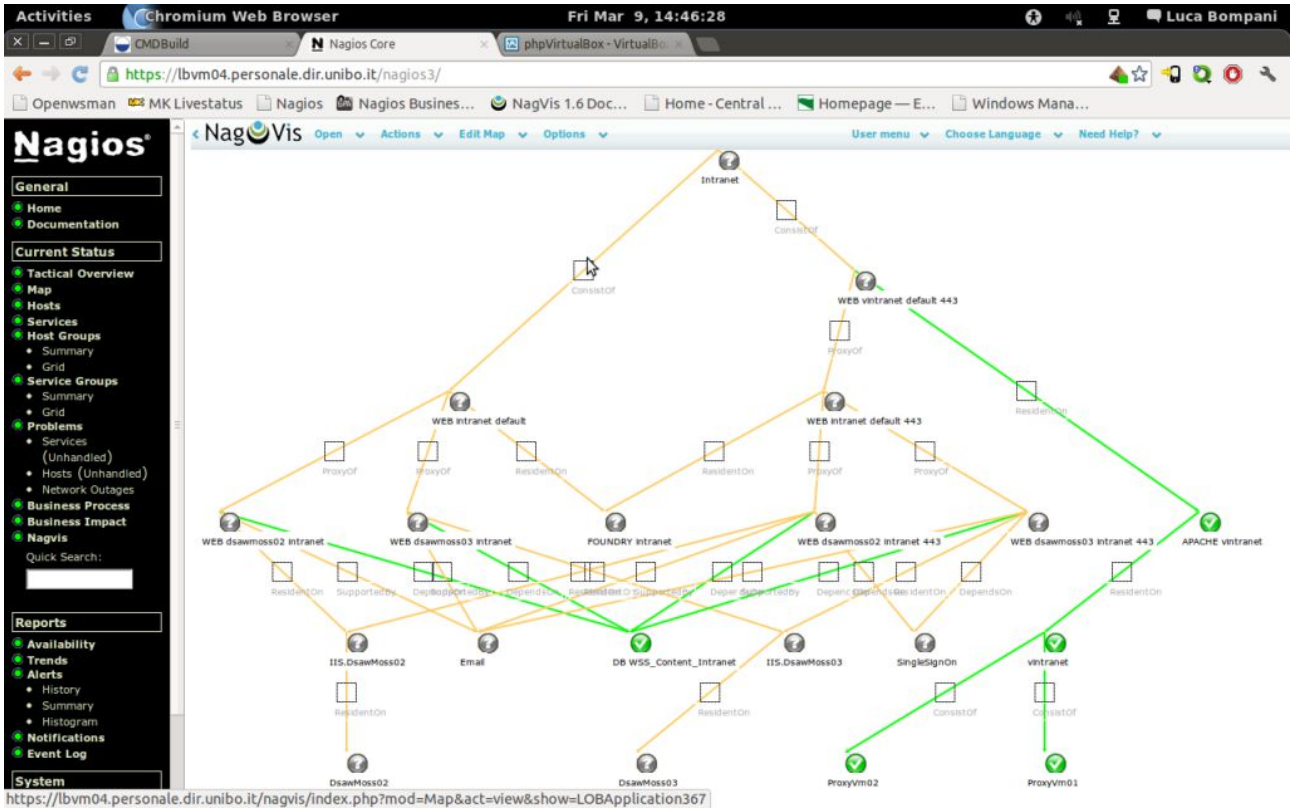
- it accesses the CMDB via a new webservice "CMDBf compliant" (http://www.dmtf.org/standards/cmdbf), which is implemented by the same contributorù
- it interprets the assets and relations contained in CMDBuild dynamically , being guided by some specially crafted metadata in the form of additional attributes on the involved classes and domains (see below)
- it creates a representation of the in-memory data in the form of a graph, by using special opensource tools
- it uses standard algorithms for the sorting and the visit of the nodes of a graph and to generate the monitoring configuration files

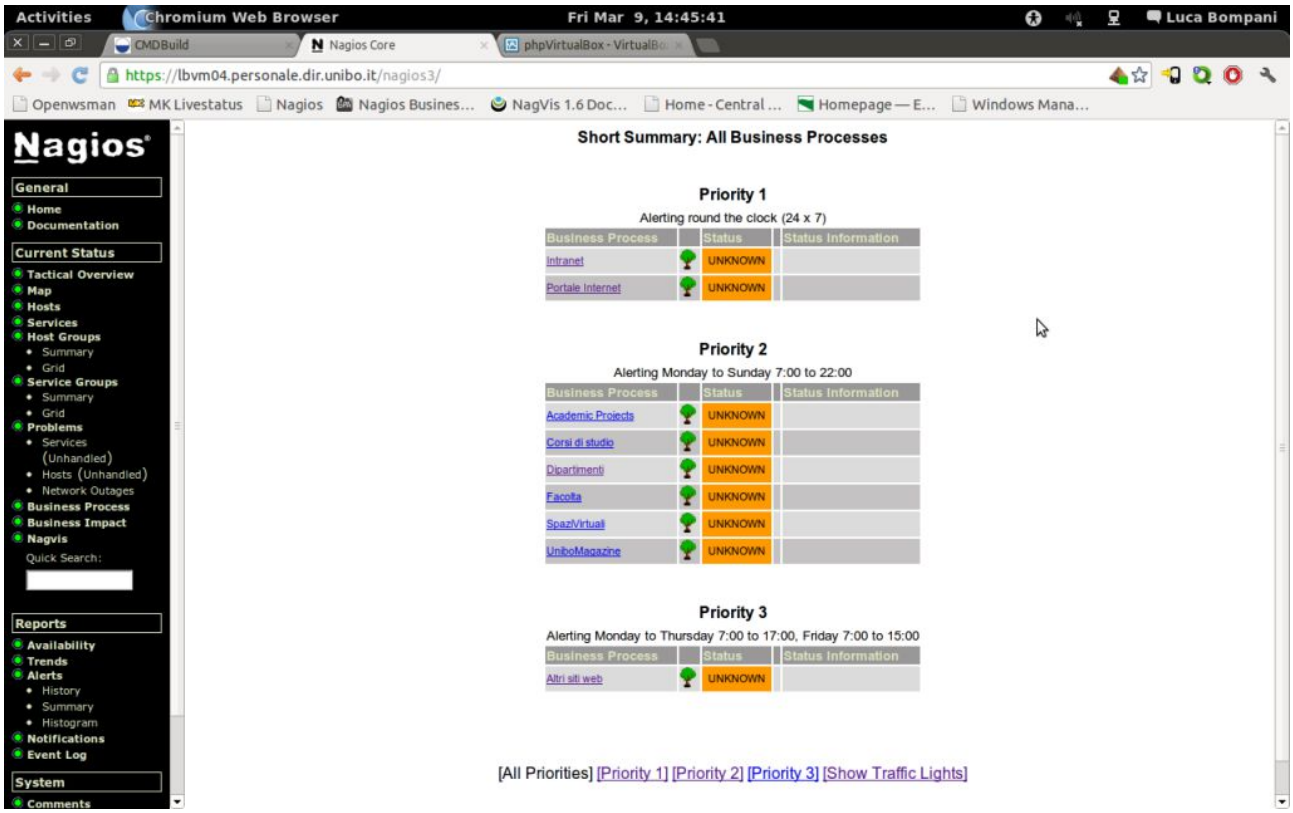
The metadata included in the CMDB in the form of additional attributes are correlated:

- to the classes
    - "host =" true "means that for that instance a Nagios host must be created
    - "serviceGroup = true" means for that instance a Nagios serviceGroup must be created
- to the domains
    - "checkAssociation= true" means that the report should be used to associate the Nagios services to the hosts
    - "ckeckDependency = true" means that the report should be used to generate the dependencies between the Nagios services
    - "ckeckPropagation = true" means that the report should be used to propagate the checks associated with an asset to another asset
    - "internalDependency = true" means that the report should be used in the generation of NagVis maps
    - "clusterAttribute = Cluster" means that that report should be used to group the relations in clusters: the check on a cluster fails only if all the cluster components fail

Three screenshots regarding, respectively, the integration with Nagios/NagVis NetEye, and NagiosBP are shown below.


Activities
Chromium Web Browser
Fri Mar 9, 14:44:54
Luca Bompani
Nagios
Current Network Status
Host Status Totals
Service Status Totals
Service Status Details For All Hosts


Activities
Chromium Web Browser
Fri Mar 9, 14:46:28
Luca Bompani
CMDBuild
Nagios Core
https://lbvm04.personale.dir.unibo.it/nagios3/
Openwsman
MK Livestatus
Nagios
Nagios Busines...
NagVis 1.6 Doc...
Home - Central ...
Homepage — E...
Windows Mana...
Nagios
General
Home
Documentation
Current Status
Tactical Overview
Map
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Business Process
Business Impact
Nagvis
Quick Search:
Reports
Availability
Trends
Alerts
History
Summary
Histogram
Notifications
Event Log
System
NagVis
Open
Actions
Edit Map
Options
User menu
Choose Language
Need Help?
Intranet
WEB vintranet default 443
WEB intranet default
WEB intranet default 443
WEB dsawmoss02 intranet
WEB dsawmoss03 intranet
FOUNDRY intranet
WEB dsawmoss02 intranet 443
WEB dsawmoss03 intranet 443
APACHE vintranet
IIS.DsawMoss02
Email
DB WSS_Content_Intranet
IIS.DsawMoss03
SingleSignOn
vintranet
DsawMoss02
DsawMoss03
ProxyVm02
ProxyVm01
https://lbvm04.personale.dir.unibo.it/nagvis/index.php?mod=Map&act=view&show=LOBApplication367


Activities
Chromium Web Browser
Fri Mar 9, 14:45:41
Luca Bompani
CMDBuild
Nagios Core
https://lbvm04.personale.dir.unibo.it/nagios3/
Openwsman
MK Livestatus
Nagios
Nagios Busines...
NagVis 1.6 Doc...
Home - Central ...
Homepage — E...
Windows Mana...
Nagios
General
Home
Documentation
Current Status
Tactical Overview
Map
Hosts
Services
Host Groups
Summary
Grid
Service Groups
Summary
Grid
Problems
Services (Unhandled)
Hosts (Unhandled)
Network Outages
Business Process
Business Impact
Nagvis
Quick Search:
Reports
Availability
Trends
Alerts
History
Summary
Histogram
Notifications
Event Log
System
Comments
Short Summary: All Business Processes
Priority 1
Alerting round the clock (24 x 7)
Business Process
Status
Status Information
Intranet
UNKNOWN
Portale Internet
UNKNOWN
Priority 2
Alerting Monday to Sunday 7:00 to 22:00
Business Process
Status
Status Information
Academic Projects
UNKNOWN
Corsi di studio
UNKNOWN
Dipartimenti
UNKNOWN
Facolta
UNKNOWN
SpaziVirtuali
UNKNOWN
UniboMagazine
UNKNOWN
Priority 3
Alerting Monday to Thursday 7:00 to 17:00, Friday 7:00 to 15:00
Business Process
Status
Status Information
Altri siti web
UNKNOWN
[All Priorities] [Priority 1] [Priority 2] [Priority 3] [Show Traffic Lights]

For further information:

The implementation is described in a case history we have presented at the "CMDBuild Day 2012", held in Bologna, at the Emilia Romagna Regional Council last May 10th 2012.

Click on this link to watch video and slides (in Italian) of the intervention “Un prototipo di integrazione fra CMDBuild e NetEye / Nagios” (A prototype of the integration between CMDBuild and NetEye / Nagios):

http://www.cmdbuild.org/it/diffusione/cmdbuild-day/cmdbuild-day-2014/2012

VCenter

General Information

VMware is one of the leading virtualization environments, while VCenter is the tool provided by VMware for the management of the infrastructure.

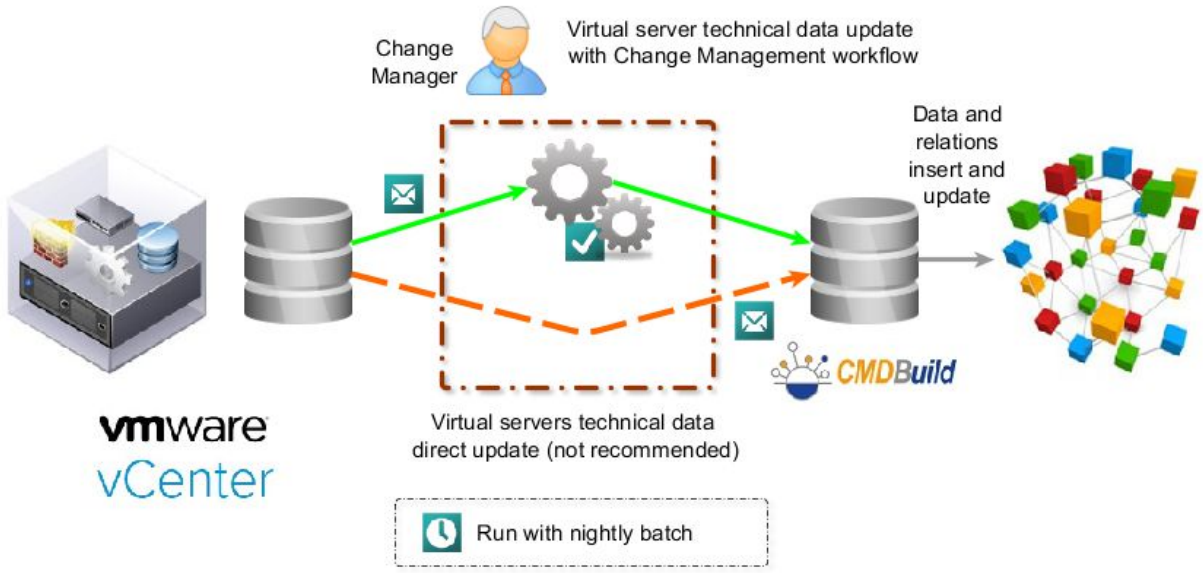
The connector described in this paragraph is therefore intended to synchronize in CMDBuild the up-to-date inventory of the virtual servers used in the VMware Infrastructure, with:

- information regarding the assigned resources (operating system, RAM, CPU, etc)
- information about the status of the server
- other custom information, managed by each organization by using the graphic interface of the application (technical coordinator, cluster, etc)

It is true that the automatic inventory tools (including OCS Inventory) are able to detect some technical information about the virtual servers, but such information is not as complete, obviously, as those provided by each individual virtualization system.

The connector allows to synchronize in CMDBuild all the virtual machines connected to VCenter, regardless of their status (on, off, etc).

The configuration of this type of connectors, that are able to communicate with external systems through the usage of API, requires the use of the Advanced Connector.



Operation modes

The interoperability with VCenter has been obtained by using the VMware VSphere API library in the connector.

The specifics of all the API methods and the descriptions of all the classes of the library are documented in this technical guide:

http://vijava.sourceforge.net/vSphereAPIDoc/ver51/ReferenceGuide/

The connector accesses to VCenter through a specific method in the API, made available from the library, and is able to recover the list of the virtual servers.

A second method of the API allows then to retrieve the detailed information available for each virtual server.

The parameters to be specified for accessing VCenter include only the IP address of the server on which it is installed and the login credentials.

It is however possible to register them in one of the configuration files of the connector, thus avoiding to have to specify them at each execution.

The detailed technical information available from VCenter are numerous and one can decide, during the connector configuration phase, which to import and which not.

The following connector, for example, synchronizes this information:

- uuid
- VM name
- VM name (dns)
- RAM size
- disk size
- Operating system
- CPU number
- IP address
- notes
- the machine state (on, off, suspended)

It is possible, by using the same library, to recover from VCenter also other custom information according to the company needs.

In our example the Connector synchronizes the following additional information:

- assignee
- informed
- cluster

The information read from VCenter APIs is then written in CMDBuild using webservices.

The syntax of the SOAP method is shown below:

```
final server = addClass(newClass("Server") //
    .withKeyAttribute("HostName") //  String
    .withAttribute("Description") // String -> HostName + ":" + Cluster.Description
    .withReferenceAttribute("Cluster", cluster) // Reference
    .withAttribute("DNSName") // TODO String
    .withAttribute("DiskSize") // Integer
    .withAttribute("OperatingSystem") // String
    .withAttribute("RAM") // Integer
    .withAttribute("CPUNumber") // Integer
    .withAttribute("Notes") // String
```

```
.withAttribute("ItemStatus") // LookUp
.withAttribute("Type") // LookUp
.withReferenceAttribute("Assignee", user) // Reference
.withReferenceAttribute("Referent", user)); // Reference
```

Archi

General Information

TOGAF [1] is an open framework to support the planning and management of enterprise information architecture, Archimate [2]is a modelling language complementary to TOGAF for the description of IT architectures, Archi[3] is the open source tool for the design of Archimate models.

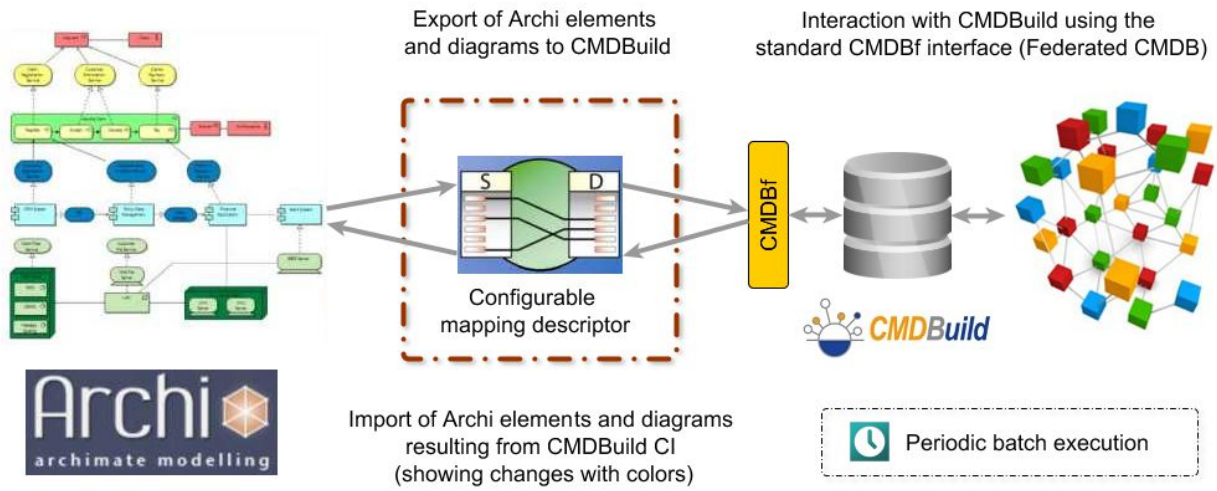
Archi allows to see a graphical overview of CI, customize colors and groupings and graphically navigate relationships, but does not have a structured database to store data.

On the other hand CMDBuild is a complete and structured repository, but does not provide a graphical view of all the infrastructure components through which analyze the impacts and graphically navigate relations.

[4]Starting from these needs two different organizations have decided to implement a plugin that synchronizes CMDBuild data into Archi, and vice versa. Each of the two organizations gained then knowledge of the work done by the other and this has led to join the efforts and release a unique product, created in partnership.

The plugin consists of a Java application directly managed by the two contributors, which can be freely downloaded at this address:

https://bitbucket.org/bompani/cmdbuild-unibo/downloads



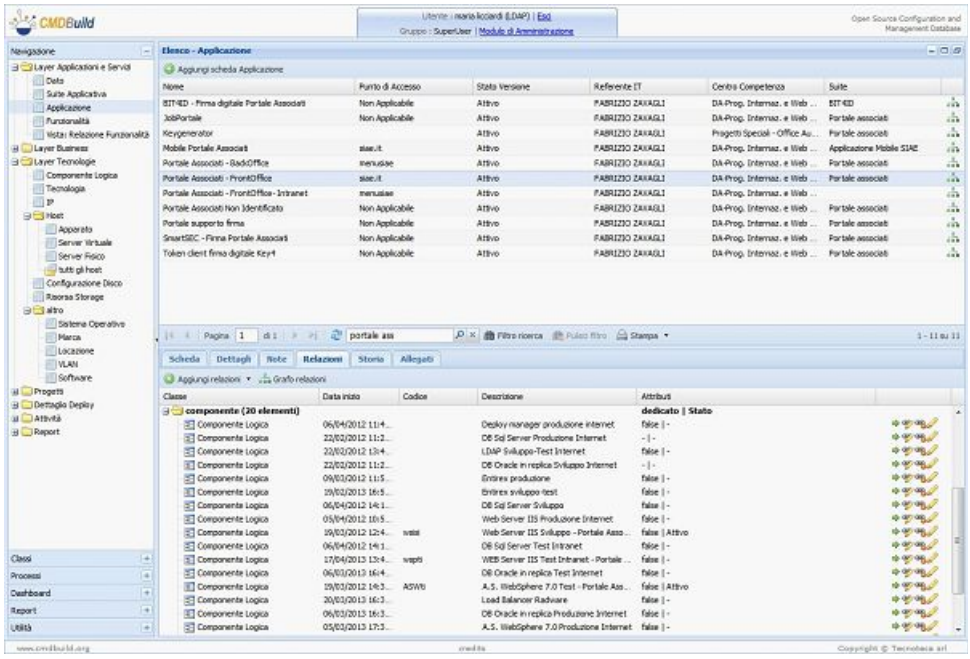
Here are two screenshots, relevant to show how the same infrastructural information are presented and made available in the two CMDBuild and Archi interfaces.

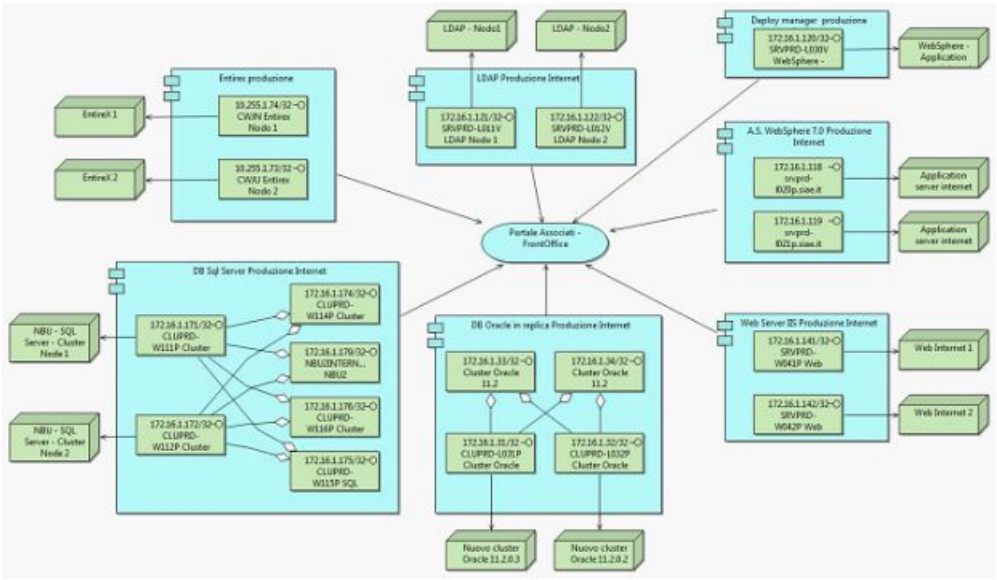
1 http://www.opengroup.org/togaf/

2 http://www.opengroup.org/subjectareas/enterprise/archimate

3 http://archi.cetis.ac.uk/

4 Bologna University and SIAE Rome





For further information:

The implementation is described in a case history we presented at the "TOGAF, Archi and CMDBuild", held in Florence in the Tuscany Regional Council last May 28, 2013.

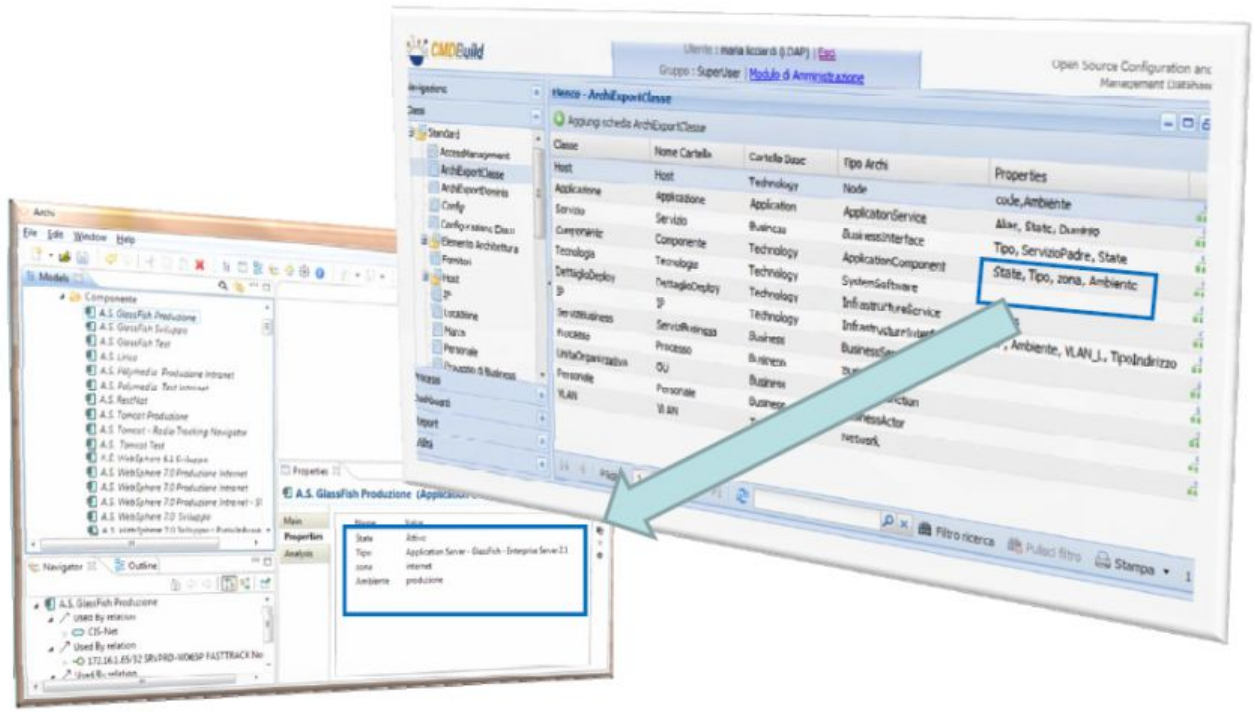
Click on this link to watch video and slides (in Italian) of the last intervention:

http://www.cmdbuild.org/it/diffusione/convegni-e-workshop/togaf-archi-e-cmdbuild

Operation modes

Also in this case the main functionality of the connector is to keep the information synchronized in two environments, defining the mapping rules between:

- Archi entities and CMDBuild classes
- Archi relations and CMDBuild domains



Main characteristics of the connector are:

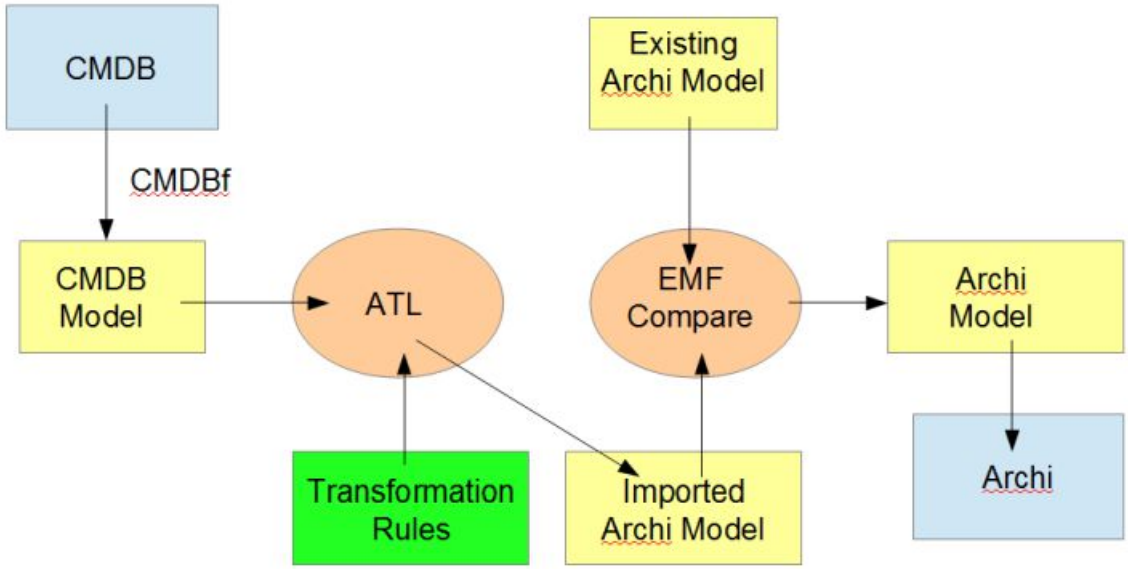
- synchronizes data and diagrams (in particular imports in CMDBuild the Archi diagrams as attachments)
- accesses CMDBuild data using a webservice CMDBf-compliant (http://www.dmtf.org/standards/cmdbf), made by one of the two connector contributors
- uses the ATL standard language to define mapping between Archi <=> CMDBuild; ATL (Atlas Transformation Language-http://www.eclipse.org/atl/) is based on EMF (Eclipse Modeling framework - http://www.eclipse.org/modeling/emf/)
- run the EMF models comparison using the Eclipse EMF Compare library
- handles bidirectional synchronization
- highlights assets that the connector would eliminate, finding them in the target system but

not in the origin

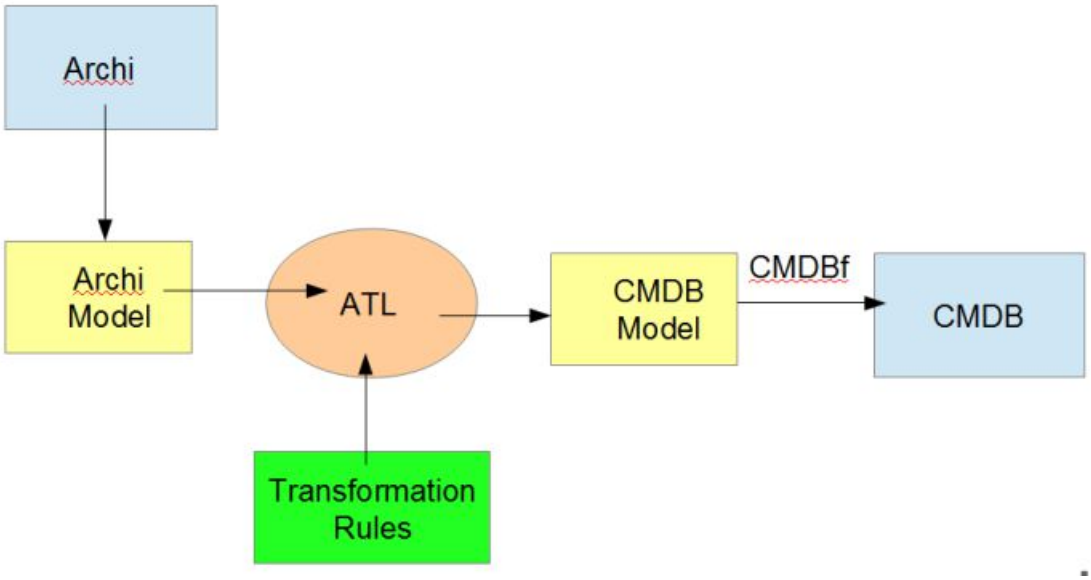
- shows the differences between the two systems

Here are the operating patterns of the two types of synchronization.

Synchronizing CMDBuild = > Archi



Synchronizing CMDBuild Archi = > CMDBuild



Configuration

Following are some guidelines for the connector configuration.

For more information please refer to the instructions in the connector's download file.

Installing the plugin

The plugin installation requires the following:

- download the archimate-cmdbuild-plugin.zip file, located in the Bitbucket repository (https://bitbucket.org/bompani/cmdbuild-unibo/downloads)
- extract the ZIP archive
- extract the plugins.zip file inside the Archi plugins directory.
- copy the file archimate-cmdb-plugin.jar to the Archi plugins folder
- import .xsd files to create the classes and the domains in CMDBuild, files are available at http://localhost/cmdbuild/services/cmdb-schema/
- restart after upload

Plugin configuration files

Archi configuration directory {HOME}/uk.ac.bolton/Archi contains the configuration files of the plugin:

- cmdbf.xml
- archi2cmdbuild.atl
- cmdbuild2archi.atl

File cmdbf.xml

Here's an example of the configuration file cmdbf.xml:

```
<configuration>
   <cmdb id="http://www.cmdbuild.org">
      <queryWSDL>
     http://localhost:8080/cmdbuild/services/soap/CMDBfQuery?wsdl
      </queryWSDL>
      <registrationWSDL>
         http://localhost:8080/cmdbuild/services/soap/CMDBfRegistration?wsdl
      </registrationWSDL>
      <query>
         ... {CMDBf Query}
      </query>
      <archi2cmdb>
         file:/home/luca/uk.ac.bolton/Archi/archi2cmdbuild.atl
      </archi2cmdb>
      <cmdb2archi>
         file:/home/luca/uk.ac.bolton/Archi/cmdbuild2archi.atl
      </cmdb2archi>
   </cmdb>
</configuration>
```

Plugin configuration

To configure the plugin:

- install the ATL SDK plugin in Eclipse
- edit files archi2cmdbuild.atl and cmdbuild2archi.atl to define the transformations between the ArchiMate meta-models and CMDB meta-models

APPENDIX: Glossary

ATTACHMENT

An attachment is a file associated to a card.

Attachments containing text (PDF, Open Office, Microsoft Word, etc.) are indexed in full text mode so that they can appear in search results.

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

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

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

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

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

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

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

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.

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 (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

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

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

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

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.

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: GUI Framework, Webservice

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

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

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

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

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

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

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

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.

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.