Version

2.2

CMDBuild®

» Webservice Manual

June 2014

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

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

Contents

Introduction

CMDBuild is an Open Source web application used to model and manage assets and services controlled by the ICT Department, and to handle related workflow operations 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.

CMDBuild provides complete support for ITIL best practices; ITIL has become a "standard de facto", it's a non-proprietary system for service management with process-oriented criteria.

With CMDBuild, the system administrator can build and extend his/her 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.

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

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.

The application 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 and perform full text searches on text-based files.

The application includes also an interface to synchronize data with external data sources (databases and mail servers); for example, you can automatically update your hardware inventory reading data from OCS Inventory - the open source computer inventory and package deployment system.

Moreover, it's possible to use the GIS feature to geo-reference and display assets on a geographical map (external map services) and / or an office plan (local GeoServer).

CMDBuild modules

The CMDBuild application includes two main modules:

- the Administration Module, used to define the data model and set config options (classes and relations, users and permissions, reports and workflows, main options and preferences)
- 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.

Available documentation

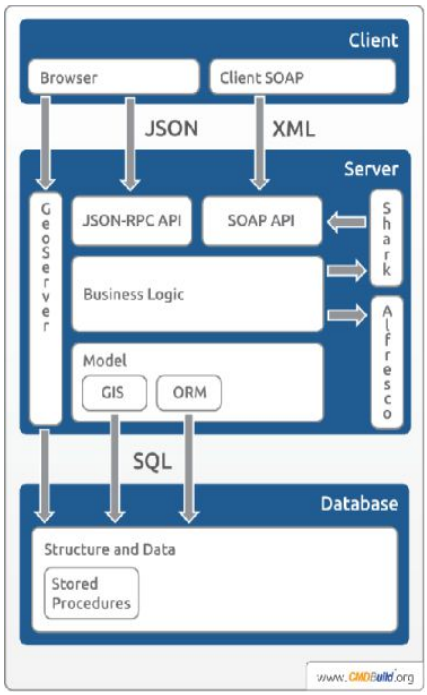
This manual is for computer technicians, who manage external applications and who are interested in interoperating with CMDBuild. It supplies the detail information necessary to implement the communication system for the read-write access of its data and functions.

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")
- connectors to sync data through external systems (“ConnectorsManual”)

Interoperability standards

Service-Oriented Architecture (SOA)



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

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

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

From a technical point of view, we chose to use the technology of webservices created with SOAP protocol.

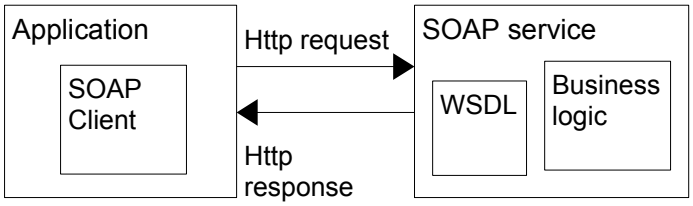
Through SOAP webservices, and safety policy permitting, CMDBuild provides the data filed in the CMDB and its management methods to allow the use within other applications involved with the information itself, both for the technical management and for administration.

Webservice and SOAP protocol

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.

SOAP is a standard protocol, based on XML, to access a webservice.



The SOAP specifications are standardized by W3C.

The webservice solution offers important architecture advantages:

- it allows to reduce the dependence between client and server applications (“weak coupling”)
- it offers an interoperability system independent from the platform and technologies
- it supports the interoperability in the web, since it is based on SOAP protocol which usually uses HTTP as basic protocol (which all firewalls enable)
- it is based on XML descriptors (WSDL)

Examples of usage

The mechanism of application interoperability, provided by CMDBuild through the SOAP webservice, can be used to activate the dialogue with every other information system pre-existing in the same organization and supporting that standard protocol.

Exhaustive examples of usage can include:

- activation in non-“JSR 168-compliant” intranet portals of simple interaction with CMDBuild for non-technicians (report print, workflow start or development, etc)
- synchronization with other CMDB tools
- integration with management applications which require to recover from CMDBuild the updated inventory of fixed assets or which must take to CMDBuild administration data of a fixed asset
- integration with monitoring technical tools which need to know information on assets subject to their control

Provided services

Method categories

The CMDBuild webservice provides methods for the external execution of basic functions managed in the system, dedicated in particular to manage and print cards and to perform processes.

In particular the available methods categories are as follows:

- management of cards: creation, modify, delete, search, history
- management of lists used to bind information fields to groups of predefined values: creation, modify, delete, search on lists and their values
- management of relations among cards: creation, modify, delete, search, history
- management of cards' attachments (filed in the repository of the document system Alfresco used by CMDBuild): upload, download, modify, delete
- management of processes configured in the system (and interpreted by the workflow engine Enhydra Shark integrated in CMDBuild): start, data recording, advancing at the following step

All methods provided in the webservice can be used upon authentication in the CMDBuild system. The authentication is performed upon the WSS Username Token profile 1.0 specification[1] with digest password.

You can find the detail description of single available methods in the next chapter.

Support data structures

Webservice methods exposed by CMDBuild use specific support data structures corresponding to the following object typologies:

- cards (data cards)
- attribute (single custom attribute in the card)
- lookup (value in the predefined list used to optimize an information attribute)
- query (filter query to select the card list)
- filter (specific to an atomic filter condition in the query)
- filterOperator (concatenation of filter conditions)
- order (organization among cards drawn from the filter query)
- relation (correlation among cards)
- attachment (document enclosed in a card)

You can find the detail description of single employed objects in the next chapter.

1 The reference document can be found at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

Authentication system

In order to have all services offered by the CMDBuild webservices, the access user must be authenticated in the system.

This is possible thanks to the authentication based on UsernameToken.

CMDBuild uses Axis2 to supply its services outside and the Apache Rampart module to manage the users authentication via webservice, employing the WSS Username Token profile 1.0 standard with digest password.

This standard provides the dispatch of credentials, meant as password username and hash. It is calculated according to the algorithm defined in the specification, in the header of the SOAP message.

Once the message has been received, the Rampart module verifies if CMDBuild provides a user corresponding to the credentials.

When this accurs and if the user is allowed to access the required service, then the webservice will report the result, otherwise there will be an error message.

Error management system

The error management in the CMDBuild system provides the use of custom error codes expressly defined.

Using the webservice, besides the errors of Axis (and the Rampart module about the authentication), also CMDBuild custom errors can be returned.

In the following table you will find the list of custom error codes in CMDBuild; potentially, they can be returned from the webservice with their meaning.

Code	***Meaning***
NOTFOUND_ERROR	Element not found
AUTH_MULTIPLE_GROUPS	The user is connected with multiple groups
AUTH_UNKNOWN_GROUP	Unknown group
AUTH_NOT_AUTHORIZED	The authorizations are not enough to perform the operation
ORM_GENERIC_ERROR	An error has occurred while reading/saving data
ORM_DUPLICATE_TABLE	There is already a class with this name
ORM_CAST_ERROR	Error in the type conversion
ORM_UNIQUE_VIOLATION	Not null constraint violated
ORM_CONTAINS_DATA	You can not delete classes or attributes of tables or domains containing data
ORM_TYPE_ERROR	Not corresponding type
ORM_ERROR_GETTING_PK	The main key can not be determined
ORM_ERROR_LOOKUP_CREATION	The lookup can not be created
ORM_ERROR_LOOKUP_MODIFY	The lookup can not be modified

Code	*Meaning*
ORM_ERROR_LOOKUP_DELETE	The lookup can not be deleted
ORM_ERROR_RELATION_CREATE	The relation can not be created
ORM_ERROR_RELATION_MODIFY	The relation can not be modified
ORM_CHANGE_LOOKUPTYPE_ERROR	The lookup type can not be changed
ORM_READ_ONLY_TABLE	Read-only table
ORM_READ_ONLY_RELATION	Read-only relation
ORM_DUPLICATE_ATTRIBUTE	There is already an attribute with this name
ORM_DOMAIN_HAS_REFERENCE	Domains with reference attributes can not be deleted
ORM_FILTER_CONFLICT	Conflict by defining the filter
ORM_AMBIGUOUS_DIRECTION	The direction relation can not be automatically determined

Description of support data structures

“Card” object

It represents a general typology of card configured in the system.

Name	Type	Mandatory	Default	Description
className	string			Class name which includes the card. It corresponds to the table name in the database.
id	integer			Card identification, it is automatically assigned by the database.
attributeList	Attribute[]			Array of "Attribute" objects containing the values of additional custom attributes in the class. They correspond to additional attributes defined in the CMDBuild Administration Module and available in the card management. The list includes also the ClassId (not the className).
beginDate	date			It shows the creation date of the current card version (in response to the initial insertion or the last change. It is a date object in the TimeZone definition format of the standard XML Schema (YYYY-MM-DDThh:mm:ssZ).
user	string			Username of the person that performed the last operation on the card.

“Attribute” object

It represents a single "custom" attribute (which is additional if compared to the two default attributes “Code” and “Description”) belonging to a card typology configured in the system.

Name	Type	Mandatory	Default	Description
name	string			Attribute name, as created with the CMDBuild Administration Module. It corresponds to the table name in the database table.
value	string			It corresponds to the attribute value.
code	string			It is developed only for "Reference" attributes with the id of the card.

“Lookup” object

It represents the value of a list with predefined values (Lookup list) used for the development of a Lookup attribute.

Name	Type	Mandatory	Default	Description
id	integer			Lookup identification, it is automatically assigned by the database
type	string			Name of the Lookup list which includes the current heading.
description	string			Description of the Lookup heading (one single heading of a Lookup list).
code	string			Code of the Lookup heading (one single heading of a Lookup list).
parent	Lookup			Lookup object corresponding to the parent heading of the current heading.
parentid	integer			Identification of the parent Lookup in the current heading (if applicable)
position	integer			Location of the Lookup heading in the related Lookup list
notes	string			Notes connected with the Lookup heading

“Query” object

It represents a filter on the values to search. The filter can be unique (Filter) or a set of conditions with a common comparison operator (FilterOperator).

Name	Type	Mandatory	Default	Description
filter	Filter			Atomic filter condition
or				
filterOperator	FilterOperator			Concatenation of filter conditions

“Filter” object

It represents an atomic filter condition to select a card list.

Name	Type	Mandatory	Default	Description
name	string			Attribute which the filter condition is applied to.
value	string			Value for the comparison with the attribute's content.
operator	string			Comparison operator (values such as EQUALS, LIKE are admitted).

“FilterOperator” object

It represents a concatenation of atomic filter conditions connected with an operator.

Name	Type	Mandatory	Default	Description
operator	string			Concatenation operator to join filter conditions (values such as AND, OR, NOT are admitted)
subquery	Filter			“Filter” object containing the filter condition applied to the query.

Sample of operation to perform*: creation of the filter “Supplier = Quasartek s.r.l. OR Supplier = IBM Italia s.p.a.”*

Filter1:

name: Supplier
value: Quasartek s.r.l.
operator: EQUALS

Filter2:

name: Supplier
value: IBM Italia s.p.a.
operator: EQUALS

FilterOperator:

subquery: [Filter1, Filter2]
operator: OR

Query:

filterOperator: FilterOperator

“Order” object

It represents the ordering standard among the cards drawn from the filter query.

Name	Type	Mandatory	Default	Description
columnName	string			Attribute which the ordering is performed on.
type	string			Ordering typology applied (only ASC and DESC values are admitted).

“Relation” object

It represents a correlation between pairs of cards, filed in the system.

Name	Type	Mandatory	Default	Description
domainName	string			Domain used for the relation.
class1Name	string			ClassName of the first card taking part in the relation.
card1Id	integer			Identifier of the first card which takes part in

				the relation
class2Name	string			ClassName of the second card which takes part in the relation.
card2Id	integer			Identifier of the second card which takes part in the relation.
status	string		A	Relation status ('A' = active, 'N' = removed)
beginDate	date			Date when the relation was created (format YYYY-MM-DDThh:mm:ssZ)
endDate	date			Date when the relation was created (format YYYY-MM-DDThh:mm:ssZ)

“Attachment” object

It represents a document enclosed in a card filed in the system.

Name	Type	Mandatory	Default	Description
category	string			Category which the attachment belongs to (from proper Lookup list).
description	string			Description related to the attachment.
filename	string			Attachment name with extension
version	string			Document version in the DMS Alfresco
author	string			User that performs the file upload
created	date			Date when the document was inserted in the DMS
modified	date			Date of the last change to the document in the DMS

Description of webservice methods

Management area of cards

Here's the list of methods dedicated to the card management.

Name	Input	Output	Description
createCard	Card card	integer id	It creates in the database a new card, containing the information inserted in the "Card" object. It returns the "id" identification attribute.
deleteCard	string className integer cardId	boolean return	It deletes logically - in the identified class - the pre-existing card with the identified "id". It returns “true” if the operation went through.
updateCard	string className integer cardId Attribute[] attributeList	boolean return	It updates a pre-existing card. It returns “true” if the operation went through.
getCard	string className integer cardId Attribute[] attributeList	Card card	It returns the required card with all attributes specified in “attributeList” (all card attributes if “attributeList” is null).
getCardList	string className Attribute[] attributeList Query queryType Order[] orderType	Card[] cardList	It returns the card list resulting from the specified query, completed with all attributes specified in “attributeList” (all card attributes if “attributeList” is null). If the query is made on a superclass, the "className" attribute of the returned Card objects contains the name of the specific subclass the card belongs to, while in the attributeList it appears the ClassId of the same subclass.
getCardHistory	string className integer cardId	Card[] cardList	It returns the list of the historicized versions of the specified card.

Management area of Lookup headings

Here's the list of methods dedicated to the management of Lookup headings (predefined values lists which a card attribute is held to).

Name	Input	Output	Description
createLookup	Lookup lookup	integer id	It creates in the database a new heading of a data Lookup list containing information inserted in the “Lookup” object. It returns the "id" identification attribute.
deleteLookup	integer lookupId	boolean return	It deletes logically the Lookup heading which shows the "id".

			It returns “true” if the operation went through.
updateLookup	Lookup lookup	boolean return	It updates the pre-existing Lookup heading. It returns “true” if the operation went through.
getLookupById	integer id	Lookup lookup	It returns the Lookup heading which shows the specified "Id" identification.
getLookupList	string type string value boolean parentList	Lookup[] lookupList	It returns a complete list of Lookup values corresponding to the specified "type". If the "value" parameter is specified, only the related heading is returned. If “parentList” takes the “True” value, it returns the complete hierarchy available for the multilevel Lookup lists.

Management area of relations

Here's the list of methods dedicated to the relation management among cards.

Name	Input	Output	Description
createRelation	Relation relation	boolean return	It creates in the database a new relation between the pair of cards specified in the "Relation" object. It returns “true” if the operation went through.
deleteRelation	Relation relation	boolean return	It deletes the existing relation between the pair of cards specified in the "Relation" object. It returns “true” if the operation went through.
getRelationList	string domain string className integer cardId	Relation[] relationList	It returns the complete list of relations of the card specified for the specified domain.
getRelationHistory	Relation relation	Relation[] relationList	It returns the relation history of a card starting from a "Relation" object in which only "Class1Name" and "Card1Id" were defined.

Management area of workflows

Here's the list of methods dedicated to the management of the processes configurable in CMDBuild.

Name	Input	Output	Description
startWorkflow	Card card boolean CompleteTask	integer id	It starts a new instance of the workflow described in the specified "Card". If the “CompleteTask” parameter takes the “true” value, the process is advanced to the following step. It returns the "id" identification attribute.
updateWorkflow	string processId Attribute[] attributeList	boolean ret	It updates the information of the card in the specified process instance.

	boolean CompleteTask		If the “CompleteTask” parameter takes the “true” value, the process is advanced to the following step. It returns “true” if the operation went through.

Management area of attachments

Here's the list of methods dedicated to the management of documents enclosed in a card.

Name	Input	Output	Description
uploadAttachment	string className integer cardId Base64Binary file string fileName string category string description	boolean return	It uploads the specified file in the DMS Alfresco and the relating connection to the CMDBuild card belonging to the “className” class and having the “id” identification. It returns “true” if the operation went through.
downloadAttachment	string className integer cardId string fileName	Base64Binary file	It returns the file enclosed in the specified card, which has the specified name.
deleteAttachment	string className integer cardId string fileName	boolean return	It removes from the DMS Alfresco the file enclosed in the specified card, which has the specified name. It returns “true” if the operation went through.
updateAttachment	string className integer cardId string fileName string description	boolean return	It updates the description of the file enclosed in the specified card, which has the specified name. It returns “true” if the operation went through.

Example of creating a “client”

In this pragraph we propose two samples of creating SOAP "clients" enabled to interface with CMDBuild 1.0

Java language

ATTENTION: we advise you to use the wsdl2java tool[2]. This tutorial implies the generation of the support classes with the above mentioned tool.

Example of suggested operation*: obtain a list of all active cards of the "Computer" class*

Once classes are generated, create the client as follows:

1. Create an instance in the ConfigurationContext class and indicate in it where the repository directory is. In the repository directory there are 2 directories: the modules directory which contains the rampart.mar file, and the conf directory which contains the file to define the safety policy which should be adopted.

```
ConfigurationContext configContext =
        ConfigurationContextFactory .createConfigurationContextFromFileSystem(
        "/path/to/repository", null);
```

2. Instance the WebservicesStub class moving in it the ConfigurationContext just created

```
        WebservicesStub stub = new WebservicesStub(configContext);
```

3. Set the authentication credentials

```
StAXOMBuilder builder = new StAXOMBuilder(/path/to/repository/conf/policy.xml);

Options options = stub._getServiceClient().getOptions();
        options.setUserName("username");
        options.setPassword("userpassword");
        options.setProperty(RampartMessageData.KEY_RAMPART_POLICY,
                                PolicyEngine.getPolicy(builder.getDocumentElement());
```

4. Instance a GetCardList object and call the server

```
        GetCardList list = new GetCardList();
        list.setClassName("Computer");
        GetCardListResponse response = stub.getCardList(list);
Card[] card = response.get_return();
```

5. At this point you can iterate on the array card content and extract the most interesting values. For example, if you want to recover the description of every Computer, the following method will be enough:

2 http://ws.apache.org/axis/java/user-guide.html#WSDL2JavaBuildingStubsSkeletonsAndDataTypesFromWSDL

```
System.out.println(card[i].getDescription());
```

PHP language

ATTENTION: since PHP doesn't support in a native way the WS-Security standard and the MTOM specification to send SOAP messages with attachments, we advise you to use a framework which can add these functionalities. For this tutorial we used the framework WSO2/PHP[3], released with Apache ver licence. 2.

***Example of suggested operation**: upload in CMDBuild a file associated to the card with Id = 13 of the "Computer" class (filing in Alfresco system)*

1. Request definition

```
$requestPayloadString = <<<XML
        <ns1:upload xmlns:ns1="http://soap.services.cmdbuild.org">
                <className>Computer</className>
                <objectid>13</objectid>
                <file><xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
        href="cid:myid1"></xop:Include></file>
                <filename>$userfile_name</filename>
                <category>$category</category>
                <description>$description</description>
        </ns1:upload>
XML;
```

2. Define a token security necessary for the authentication

```
$security_options = array("useUsernameToken" => TRUE );
$policy = new WSPolicy(array("security" => $security_options));
$security_token = new WSSecurityToken(array("user" => "username",
                                            "password" => "password",
                                            "passwordType" => "Digest"));
```

3. Instance a WSClient object to execute the request. You have to specify the MTOM use for the file transfer

```
$client = new WSClient(array("useMTOM" => TRUE,
                                            "useWSA" => TRUE,
                                        "policy" => $policy,
                                    "securityToken" => $security_token));
```

4. Store the file content in a single string using the function file_get_contents()

```
$file = file_get_contents($_FILES['userfile']['tmp_name']);
```

5. Instance a WSMessage object and execute the request

3 You can find the official website at the following address: http://wso2.org/projects/wsf/php

```
$url = "http://localhost:8080/cmdbuild/services/soap/FileTransfer";
$requestMessage = new WSMessage($requestPayloadString,
                                        array("to"=> $url,
                                                "attachments" => array("myid1" => $file)));
$response = $client->request($requestMessage);
```

Definition of authentication policy file

File structure

The policy file - which must be passed to the Rampart module for client side - must be defined using the Web Services Security Policy Language (WS-SecurityPolicy).

This language, derived from XML, allows to define security rules respected by the application.

For example, in order to define the authentication use through Username Token with Password Digest, you have to define the policy as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:Id="UTOverTransport"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ssecurity-utility-1.0.xsd"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   <wsp:ExactlyOne>
      <sp:SignedSupportingTokens
            xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
           <wsp:Policy>
             <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                  <wsp:Policy>
                    <sp:HashPassword/>
                  </wsp:Policy>
             </sp:UsernameToken>
           </wsp:Policy>
      </sp:SignedSupportingTokens>
   </wsp:ExactlyOne>
</wsp:Policy>
```

You can also indicate in the policy file, certain configuration parameters of the Rampart module.

For example, if you don't want to directly transfer the password in the code, but gather it from a database, you can suggest Rampart to use a custom class which implements the CallbackHandler function in which you can define how to find the password.

In that case you have to modify the policy file in the following way.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:Id="UTOverTransport"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-ssecurity-utility-1.0.xsd"
```

```
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   <wsp:ExactlyOne>
      <sp:SignedSupportingTokens xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
            <wsp:Policy>
               <sp:UsernameToken sp:IncludeToken="http://docs.oasis-open.org/ws-sx/ws-
securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                     <wsp:Policy>
                       <sp:HashPassword/>
                     </wsp:Policy>
               </sp:UsernameToken>
            </wsp:Policy>
      </sp:SignedSupportingTokens>
         <ramp:RampartConfigxmlns:ramp="http://ws.apache.org/rampart/policy">
            <ramp:passwordCallbackClass>
                                    org.cmdbuild.services.soap.secure.MyPasswordHandler
                           </ramp:passwordCallbackClass>
            </ramp:RampartConfig>
   </wsp:ExactlyOne>
</wsp:Policy>
```

For a deeper treatement of WS-SecurityPolicy, please refer to the related official document[4].

4 http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf

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 they can appear in search results.

WORKFLOW STEP

A process step is a process which can have one or more steps.

A step has a name, an executor, a type, attributes - if any - and methods with statements (CMDBuild API) to be executed.

A process instance is a single process that's been automatically activated by the application or manually activated 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 describes 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's one of the major ITIL processes managed by the application.

See also: CI, ITIL

DASHBOARD

In CMDBuild, a dasboard 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 softwares 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's 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

ITIL

"Best practices" system that established a "standard de facto"; it's 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's possible to define new "LookUp" tables according to organization needs.

PROCESS

The term "process" 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: Worflow 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, ie 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.

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.