

SAPID CMF

framework to develop content-dependent rich web-applications

Manual

Contents

Contents	1
Preface	6
Introduction to SAPID CMF	7
What is SAPID CMF	7
Why SAPID CMF?	7
SAPID CMF History	7
Basic Concepts	8
Data Organization within the Framework	8
Information Objects	8
Documents	8
Records	9
Files	10
Information Organization Objects	10
Structure	10
Environment Variables	10
Templates	11
Field sets	11
Functionality Scripts	12
Record Lists	12
Users	12
Reports	13
Configuration	13
Platform Architecture	15
DB Architecture	15
Platform DB	15
Additional Tables	15
Component Model	16
Installation and Update of SAPID CMF	18
Installation of SAPID CMF on a remote server	18

Installation of SAPID CMF on your local computer.....	18
Update of SAPID CMF	20
Project Creation	21
Resource Creation	21
Structure Creation	21
Templates Creation	22
Functionality Script Creation.....	23
Web-project Administration	24
Administrative Panel.....	24
Structure Section	24
Services Section	32
Users	37
Reports	38
Config Section	39
Inline Administrative Mode	40
Project Debugging	43
Framework Adaptation	44
Plugin Creation	44
Defined Events	45
Programm Code of Adaption	45
\$env Structure	46
Common Functions.....	47
Third-party's Visual Editor Using (WYSIWYG)	48
FCKEditor Using	48
Administrative Panel Adaptation.....	49
Creation of Administrative Panel Presentation Themes.....	51
Administrative Application Creation	52
Administrative Application Structure.....	52
How to Build Application.....	53
Creation of Applications with Lists.....	55

How to Make Lists Manageable	57
SAPID CMF API.....	62
ADO API (DB Control Interface)	62
All()	62
One()	62
Update().....	62
lastNumRows()	62
lastAffected().....	63
lastInsertId().....	63
StartTransaction()/CompleteTransaction().....	63
Prepare().....	63
DM API (Document Management Interface).....	63
Document Definition	64
Document Attributes	64
addSite()	65
add()	65
update()	65
delete()	66
updateData().....	66
copy().....	66
get()	67
getByCondition()	67
getList().....	67
getData().....	68
getSiteID().....	68
identify()	68
setPermission().....	69
backup().....	69
restoreBackup()	69
BackupList().....	69

RM API (Record Management Interface)	70
Record Definition	70
Record Attributes	70
add ()	71
update()	71
delete()	72
updateData().....	72
get()	72
getList().....	73
getByCondition()	73
getData().....	73
copy().....	73
copyList()	74
addIndex().....	74
deleteIndex().....	74
setPermission().....	75
backup().....	75
UM API (User Management Interface)	75
get ()	75
update ()	76
delete ().....	76
getData ().....	76
updateData ().....	76
deleteData ()	77
getByLogin()	77
RD API (Request Dispatcher).....	77
clean ().....	78
set()	78
create ()	78
Developer Handbook	79

Environment Variables	79
Content Queries (QC)	80
Expressions	81
CMS – applications	82
get_infochannel()	82
pagination()	84
get_tree()	85
get_track()	86
get_search ()	86
get_ad ()	87
Appendix	88
Support	88
Community site: Developer Handbook	89
Environment Variables	89
Content Queries (QC)	90
Expressions	91
CMS – applications	92
get_infochannel()	92
pagination()	94
get_tree()	95
get_track()	96
get_search ()	96
get_ad ()	97
Appendix	98
Support	98

Preface

This manual is written for people who want to build informational systems based-on Web and content-dependent applications with SAPID CMF.

The manual contains common information of the framework, its structure, basics to develop informational systems based-on SAPID CMF and SAPID CMF API Reference to adapt the framework for special needs (See Adaptation Section).

This manual expects a basic knowledge of HTML, XML, PHP, SQL, as well as experience to work with an FTP-client. Before you begin to read the manual, it would be useful to learn “XML Sapiens 2.0 Specification” (<http://xmlsapiens.org>). A familiarity with the Model-View-Controller programming pattern is helpful too.

SAPID CMF is a free software. You don't need to pay for it and you can use it as you wish. This is an Open Source, what means you have full access to its source code. The last version of the framework you can download at site <http://sapidcmf.sourceforge.net>

Introduction to SAPID CMF

What is SAPID CMF

SAPID CMF is an open source platform to develop content-dependent web applications. It's a major Content Management System (CMS), which can be used for site development and this's a framework allowing unlimited CMS development in accord your growing-up needs.

SAPID CMF contains library set, classes and run-time tool kit for programmers, who build web-applications. SAPID CMF component model relies on MVC pattern. SAPID CMF organizes data, its presentation and functionality based-on XML Sapiens concepts.

Why SAPID CMF?

- Flexible model to pattern data structures, presentation and functionality based-on XML Sapiens 2.0 markup language, what provides broad possibilities of once-made work reusing.
- It's optimized to build applications of Web 2.0 Generation
- It contains AJAX-based graphic user interfaces and set of ready widgets (tree, grid, forms, panels and so on)
- It uses the Model-View-Controller (MVC) pattern
- It's ready to evolve (plugins, decoration themes, aspect-oriented event model)
- It's compatible with PHP4 and PHP5
- It verifies data on the layer of customizable queries (QC)
- It has flexible and effective templating (XML Sapiens)
- It implies manageable access control lists

SAPID CMF History

XML Sapiens markup language has been invented and documented by Dmitry Sheiko (<http://cmsdevelopment.com>) Red Graphic Systems programmer in 2003 to find an optimal model to separate data, presentation and functionality for Site Sapiens content management system (www.sitesapiens.com).

In 2004 Red Graphic Systems company proclaimed XML Sapiens specification. SAPID content management system was created by Max Barishnikov to demonstrate XML Sapiens features. Since SAPID CMS is very simple, even doesn't demand DB and allows INLINE administration (what you see is what you edit) the CMS got popular. However SAPID CMS was not projected as a system to build serious web-solutions, but just to demonstrate XML Sapiens features. After 3 years since the first version of SAPID was presented SAPID Content Management Framework appeared. The framework is meant to build content-dependent web-applications, to create informational systems based-on Web. Its key difference from SAPID CMF is ability to evolve without limit. Attainment of this aim got possible thanks to using such technologies as aspect-oriented event model (AOSD), plugins and XML Sapiens 2.0. SAPID CMF contains lightweight AJAX-framework, what provides growth of graphical user interfaces and services.

Basic Concepts

Data Organization within the Framework

Informational web-space is whole of all sites, managed centrally, by one system. All data of informational web-space based-on SAPID CMF can be presented as objects of 3 types: documents, records and files.

Any of those objects can have attributes, content, relations and permissions. Documents and records can include inner record list.

Documents of informational web-space can be organized by hierarchical structure.

Document templates serve to assign decoration and document content structure. Templates can include each other.

PHP-scripts, as a rule, are included in templates and DDC and serve as page controllers.

Field sets are meant to assign record structure and user and file description profiles.

Functionality scripts (DDC) are applied to create functional bars. You can learn more of DDC in "XML Sapiens 2.0 Specification" document.

Information Objects

Documents

Document is an informational object of resource hierarchical structure tree. Document can be a section of the structure. Document can contain inner record list.

DDC and `get_tree()` CMS-application usually are used to get document list, to form navigational menu (See "XML Sapiens 2.0 Specification").

Document attributes

Name – document definition within the structure;

Template – presentation template of the document;

Document visibility – this defines if the document is visible in the structure;

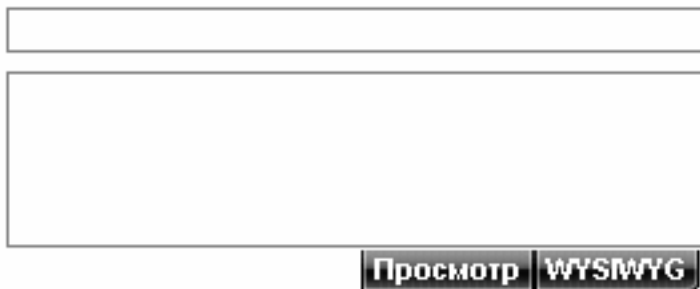
Tags – key words to assign associative relations between documents.

Content

Document content is defined by content queries, presented in the document template and included templates. Thus, if you assign a new template to a document, tab of the page management interface will have new set of content query fields. Content query within template is defined by XML Sapiens construction `<sapi:apply name="qc.name.value" />`. Caption of the query is defined by title attribute of `sapi:apply` element. Type of query is assigned by attribute type. For example, you should use following construction to get query set

Some title

Text



```
<sapi:apply name="qc.mytitlevar.value" type="inputtext" title="Some title" />
<sapi:apply name="qc.mybodyvar.value" type="article" title="Text" />
```

Content query types are defined in files of the same name, located in folder: /views/default/qcs/.

Permissions

On default users of administrative groups in SAPID CMF have permissions to read and write into a new document. Unauthorized users has permissions only to read. However there is a possibility to set certain permissions for different user groups.

Tab “Security” of document property management window in administrative user panel serves to manage document access permissions. Use Doc class (see DM API Section) to manage document access permissions.

Records

Record is an informational object of lineal lists of the resource (site). Content of a record, like document's content, caused by specified field set. Records are united by lists, which can contain lists as well.

DDC and get_infochannel() CMS-application usually are used to get record list, news, catalogues and galleries (See “XML Sapiens Specification”).

Record Attributes

Name – record definition within the structure;

Field set – set of content query fields of the record;

Tags – key words to assign associative relations between records.

Content

Record content is defined by content queries, presented in the field set. Files of field sets are located in folder /views/delivery/fieldsets/. Content query of field set is defined by XML Sapiens language construction `<sapi:apply name="qc.name.value" />`. Query caption is defined by title attribute of sapi:apply element. Type of query is defined by type attribute.

Content queries types are defined in files of the same name, located in folder /views/default/qcs/.

argsstringwithoutslash – the same, but without slashes (for example, gallery);

admin_view_http_path – address of administrative panel templates root folder (for example, <http://sapidcmf.rh8.rg/views/default/>);

delivery_view_http_path – address of site templates root folder (for example, <http://sapidcmf.rh8.rg/views/delivery/>);

http_path – current site address (for example, <http://sapidcmf.rh8.rg/>);

admin_http_path – administrative panel address (for example, <http://sapidcmf.rh8.rg/admin/>);

system_configuration_date – framework configuration date (for example, 2007/08/22);

system_version – framework version (for example, Build 85);

user_id – authorized user identifier (for example, pass);

site_id – open site identifier (for example, 1);

record_id – requested record identifier. If record is not requested it returns 0;

user_fullname – full name of authorized user;

user_login – login of authorized user;

user_profile – description template of authorized user (for example, default.tpl);

document_title – title of the open document (for example, Gallery);

document_template – template of the open document (for example, gallery.tpl);

document_id - identifier of the open document (for example, 9);

document_data_x – content data, wherein x is name of a content field;

record_data_x – content record, wherein x is name of a content field.

Templates

Templates defines how the site page will look. Template assigns presentation (HTML, RSS and so on). It uses XML Sapiens language to define content structure (QC), functionality (DDC), for environment variable display. Effective architecture means templates for all iterative site page bars of informational system, included within templates these pages. Thus if something is changed in template with Copyright line, this change will be reflected in all pages with this template.

Files of templates are located in folder /views/delivery/templates/.

Field sets

Field set serves to define record structure. Content queries in field set file point what kind of content will be requested for specified record in administrative panel and what kind of content will be shown on site pages. Field sets can be defined for different states of record. The same record can be displayed in common list on the site for one configuration and when page is chosen for the other configuration. Record output is caused by functionality script (DDC). Record presentation can be described in DDC by means content variables assignment

(&this.this.имя_поля_QC.value;) or due field set request for specified state <sapi:include href="*field_set_file*" parse="fieldset" state="state_A" />.

Field set admits code of content presentation.

Sample of field set:

```
<?xml version="1.0"?>
<sapi version="2.0" xmlns:sapi="http://www.xmlsapiens.org/spec/sapi.dtd">
  <sapi:fieldset type="default" state="admin" title="">
    <sapi:body>
      <sapi:apply name="qc.x1.value" type="type1" />
    </sapi:body>
  </sapi:fieldset>
  <sapi:fieldset type="default" state="delivery" title="">
    <sapi:body>
      <sapi:apply name="qc.x2.value" type="type2" />
    </sapi:body>
  </sapi:fieldset>
</sapi>
```

Files of field sets are located in folder /views/delivery/fieldsets/.

Functionality Scripts

Functionality scripts (DDC) serve to describe behavior of site dynamical bars such as record list (news, catalogues, galleries and so on). You can learn more of DDC in XML Sapiens specification. How to include functionality scripts is described in document templates or in other DDCs.

Forms of solutions based-on AJAX need controller to transfer data on the server. You can assign controller in document template or in DDC body with construction <sapi:include href="*script_name*" parse="php" />.

Files of functionality scripts are located in folder /views/delivery/ddcs/.

Record Lists

Document or record can contain a record list. A record can contain records. You can use DDC to display record list on sites and getinfochannel CMS-application. You should use Rec library to control record lists for CMS program adaptation.

Users

User object of the framework contains following attributes:

Name – identifier of the user. Usually it coincides with login;

Login – user's login;

Password – user's password;

Name – the name of common list;

Email – user's email address;

Language – interface language of the user;

Activity state – user activity trigger;

Profile – name of user profile field set. Files of user profile field sets are located in folder /views/delivery/templates/users/.

Users can be united in groups. User groups can be united in groups as well. You should use User class to control user accounts for framework adaptation.

Access permissions can be delegated to a user of the framework. Access permissions to an administrative application can be delegated to a user group in ACL-file of the application. For example, app/structure/app.db.

Syntax:

```
<?xml version="1.0" encoding="utf-8" ?>
<treeitem titlekey="Structure" sortkey="1">
    <acl>
        <group name="developers" permissions="7" />
    </acl>
</treeitem>
```

ACL element means a section to have access to the application specified in treeitem element.

GROUP element defines access to the application for the group and contain following attributes:

name – identifier of a group or a user;

permissions – access permissions in binary system (0-1 – no access, 2-3 – access to change, 4-7 – access to read, 6-7 – access to read and change).

Reports

Any operations with APIU or actions in administrative panel is reflected in "System event report".

Configuration

Basic configuration of the framework is designated in file file /config/rc.conf.php.

HTTP_PATH – site web-address on default;

SITEID – site ID on default;

DEFAULT_LANGUAGE – interface language on default;

ROOT_PATH – full root address on the server;

ADMIN_EMAIL – email address for feedback on default;

VERSION – SAPID CMF version;

CONFIGURED - date of framework installation;

DBPREFIX – prefix of DB tables;

ADMINAREA_PREFIX – administrative panel virtual folder;

CONTROLAREA_PREFIX – virtual folder of framework AJAX-controllers;

TOTALCACHING – on/off page caching;

VENDORS – address of third-party solutions folder;

DEBUG – trace mode (0/1/2);

DOCCOPIESMAXNUMBER – number of all kept document/record content versions;

DOCCOPIESMAXNUMBERPERUSER – number of kept document/record content versions for each user;

BACKUPING - flag to keep document/record content versions or not;

\$GLOBALS["DBConnect"] – DB access data;

SOAP_SERVER_URL – SOAP-server address;

FILEDATEFORMAT – file date format on default;

INLINEMODE – on/off for INLINE mode availability;

THEME – current theme of administrative panel presentation;

PAGINATIONFRAMELENGTH – pagination frame length;

DBTABLESSTRUCTSPATH – location of DB restoring scripts;

GMTOFFSET – time zone offset;

Additional configuration can be designated in “Config” section of administrative panel. These data will be available in templates and DDC as environment variables `&config_variable.value`; and you can use them within adaptation (in plugins) as `&env->Value["config_variable"]`.

Platform Architecture

DB Architecture

Platform DB

_config – register of configuration data;

_doc_data – document content;

_doc_data_bak – document content versions;

_doc_structure – document structure;

_log – administrative events report;

_permissions – access control list;

_rec_data – record content;

_rec_data_bak – record content versions;

_rec_structure – record structure;

_tagcloud_index – structure tags indexes;

_user_accounts – user structure;

_user_profilesdata – user profile data;

_user_settings – user GUI-customization data;

_whole_enums_data – enumeration data (data of common form of SELECT QCs).

Additional Tables

_advertising – advertising bars account data;

_maillist – maillistings configuration;

_rec_rate – record rate data;

_thesaurus – content thesaurus data;

Component Model

APP – business logic and controllers of administrative panel applications

CACHE – cache files

CONFIG – configuration

DOCUMENTATION - documentation

INSTALL – application of framework deployment

LANGS – interface language files

LIBS – libraries

- CMSAPPS - CMS-applications of the framework

- MODEL

 - DBO – libraries to interact with DB abstract layers

 - Php4 – model libraries for PHP4

 - Php5– model libraries for PHP5

- VIEW

 - PHP4 – view libraries for PHP4

 - PROCESSOR –XML Sapiens 2.0 processor libraries

 - PHP5 – view libraries for PHP5

 - PROCESSOR– XML Sapiens 2.0 processor libraries

PLUGINS – framework adaptation

TESTS – unit tests

VENDORS – third-party libraries

- adodb_lite

- fckeditor

- graphic_snips

- jscalendar

- lastRSS

- mmplayers

- simple_openid

- simpletest

tinymce

yui

VIEWS – presentation templates and functionality patterns

 DEFAULT – templates and patterns of administrative panel applications, DEFAULT theme

 CSS – styles

 DDCS – functionality scripts

 FIELDSETS – field sets

 JS –JS scripts

 PIC - images

 QCS – content query patterns

 TEMPLATES – administrative panel application templates

DELIVERY – site templates and patterns

 CSS – styles

 DDCS – functionality scripts

 FIELDSETS – field sets

 JS – JS scripts

 FILES – files to be downloaded from the sites

 IMG – images

 QCS – content query patterns

 TEMPLATES – site page templates

 THUMBNAILS – images thumbnails

WIDGETS – site widgets

Installation and Update of SAPID CMF

You need distributive pack of the framework to install SAPID CMF, which you can download at <http://sapidcmf.sourceforge.net>.

Installation of SAPID CMF on a remote server

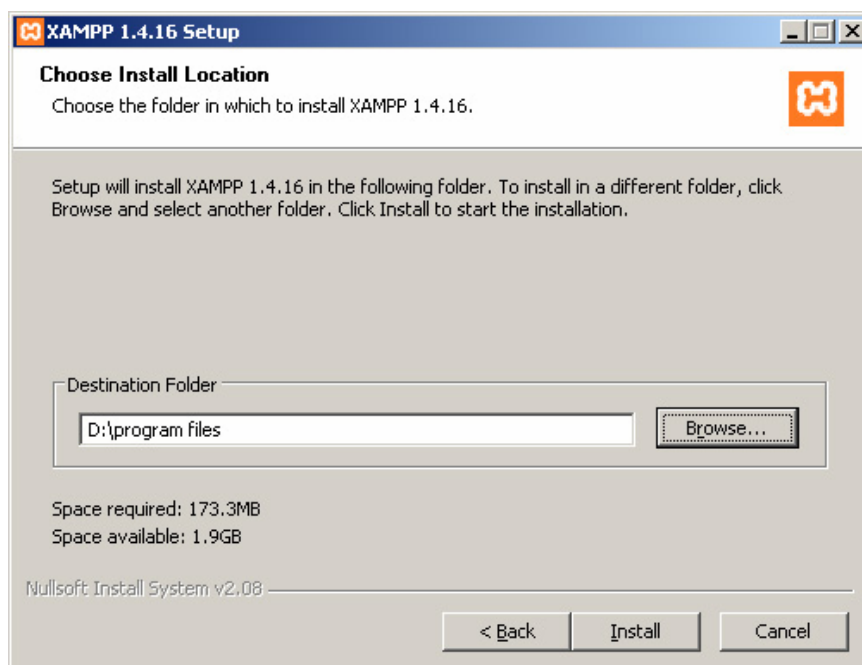
You can order hosting plan for PHP4 or PHP5 from any hosting company. Be sure it includes DB MySQL 4 or 5 and Apache Rewrite Module. You also will need PHP extension library mbstring (a library to use strings with multibyte/UTF8 characters)

Use Total Commander (www.ghisler.com) or any other FTP-client to copy files of the distributive pack into folder html (it means the root folder of the web-project) on your server. Then you should to set up permissions to folders cache, cache/ddc, tmp, views/delivery/files, views/delivery/img, config/rc.conf.php – 777 (full access). In the case of Total Commander you can choose folder of file and go to application menu Files/Change Attributes and mark all checkboxes or enter 777 in the field of access mask.

Next you can launch web-browser (for example, FireFox) and type in address line the name of your site (say, myownstartup.myhost.com). You will see the window of SAPID CMF installation application. See “SAPID CMF Installation Application” section to get known what to do farther.

Installation of SAPID CMF on your local computer

If you want to install SAPID CMF to your local computer, use XAMPP Windows tool kit. It will setup you apache server on your computer automatically.

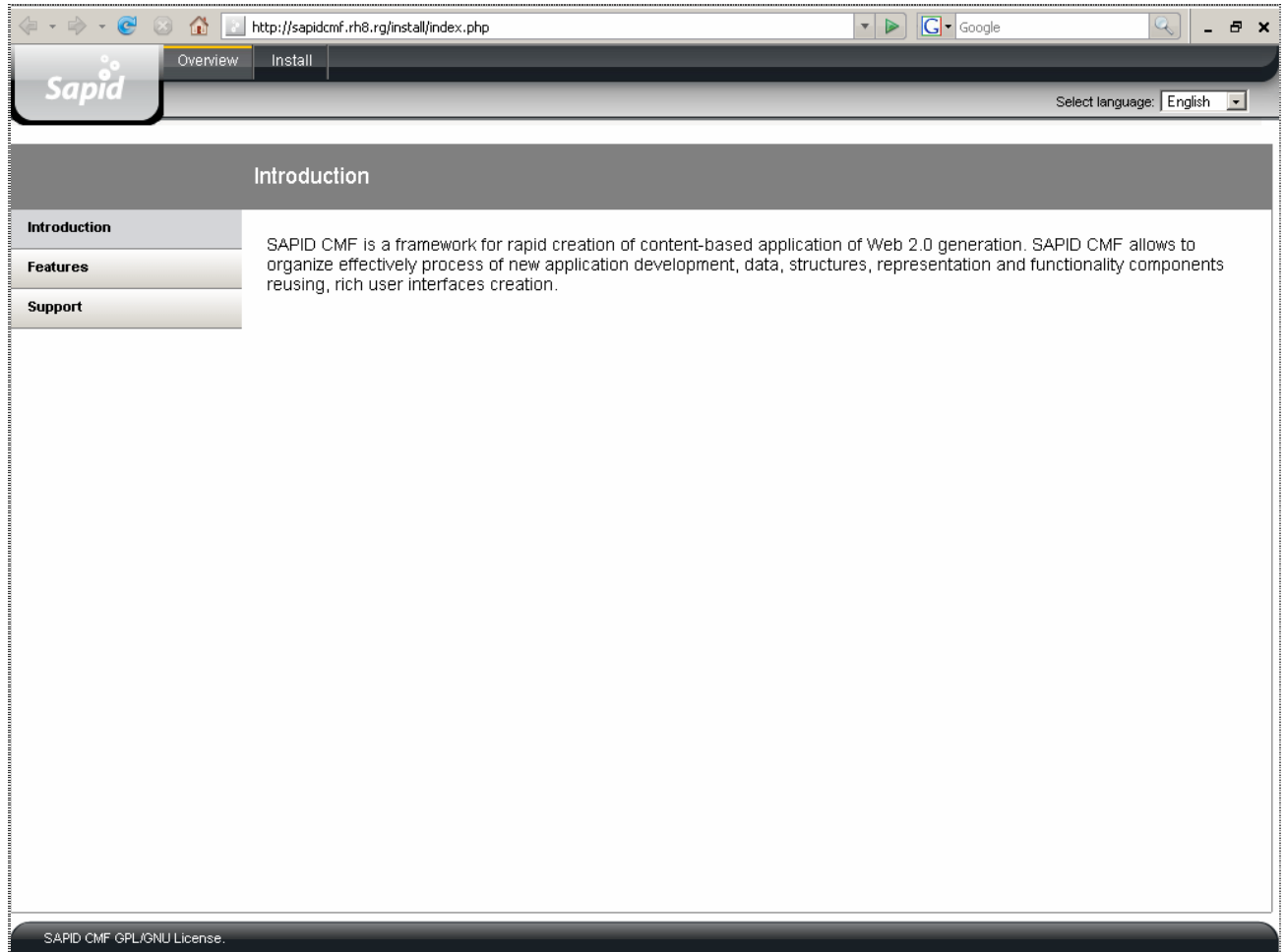


You can see how to use XAMPP and download this free at <http://www.apachefriends.org/en/xampp-windows.html>

When you finish XAMPP installation you will get folder \xampp\htdocs. You can create your folder sapidcmf.rg there and unpack the distributive pack into the folder. When you type in you browser address line sapidcmf.rg you will see the window of SAPID CMF installation application. See “SAPID CMF Installation Application” section to get known what to do farther.

SAPID CMF Installation Application

When you start the application the first time, you will see following screen:



Use menu "Install" to start install. You will see "Welcome" page wherein you can press the button "Process to next step". The application will tell you if you have all necessary options installed on you server. Then when you go to the next step ("Start install" button) you will be requested for server DB connection data.

If PHP5 is installed on your server it's likely you will see PDO in the list of available libraries of DB objects. That is native PHP library to manage DB objects and obviously you project will work quickly with that library. Then you should enter DB connection data. If you use a remote server, your hosting company has to tell you these data. If you are setting up the framework on you local computer under Windows OS, you can leave default values in the fields without changes. But this case you ought to create DB manually. It can has name sapidcmf. So when you have DB created you will enter its name in the field "Data base name".

On the next step you will be requested with questions to configure your project on SAPID CMF. Don't forget to choose your native language for administrative panel and login/password to have access into administrative panel.

Update of SAPID CMF

In order to update framework version you should download pack with update files at <http://sapidcmf.sourceforge.net> and unpack it over files of your project. The pack of update hasn't templates and patterns of delivery side. So it can't do any damage to your work in the project.

If the pack contains update.php script in the root folder. Start the script after pack files are uploaded on the server and follow its instructions.

Project Creation

Resource Creation

First of all, enter to administrative panel of framework (<http://yousite.com/admin/>). You has to see a screen with structure tree. Now you should press link against top position ("Informational Network") and enter name of the resource.

Than it's demanded to hover mouse cursor over icon of created resource. You will find ID of created resource in tooltip (for example, #1). Write it into configuration file `config/rc.conf.php`

```
if(ereg("lang.site.com", $_SERVER["HTTP_HOST"])) {  
    define("HTTP_PATH", "http://lang.site.com/");  
    define("SITEID", 59);  
} else {  
    define("HTTP_PATH", "http://sapidcmf.rh8.rg/");  
    define("SITEID", 1);  
}
```

In this case sub domain `lang.site.com` is related to the resource with ID=59 and sub domain `sapidcmf.rh8.rg` to the resource with ID=1.

Structure Creation

You can use administrative panel to cerate a new resource or you can restore the structure from a text file. In the last case you ought to prepare file in the format:

```
@site:Name of new resource  
Number of level; document name; address variable;  
Number of level; document name; address variable;  
...
```

or

```
@site:Name of new resource  
Number of level; document name; address variable; template file name;  
Number of level; document name; address variable; template file name;  
...
```

or

```
@site:Name of new resource  
Document name; address variable;  
    Document name; address variable;  
        Document name; address variable;
```

...

Wherein number of indents (tabs) defines nesting.

Example:

```
@site:Sample Site
1;Main page;var1;demomain.tpl
    Catalogue;var2;channel.tpl;
        Doc 111
            Doc 12
            Doc 13
Doc 2
    Doc 21
    Doc 22
        Doc 221
            Doc 2221
                Doc 22211
                Doc 22212
                Doc 22213
                Doc 22214
                Doc 22215
                Doc 22216
                Doc 22217
                Doc 22218
                Doc 22219
                Doc 22220
                Doc 22221
                Doc 22222
                Doc 22223
```

Copy this file to TMP folder and go to the System Console application of the administrative panel (Services section). Activate the console with service command "command" and use command:

```
import /tmp/your_file_name
```

Templates Creation

You need approved sketches of site typical pages and their HTML-code to be ready to create templates on the framework. If you have that, you can analyze sketches to find common bars and functional bars. HTML-code of common bars is located as a file in the template folder (usually it's a nested folder of /views/delivery/templates/, for example, subtemplates). HTML code of functionality bars shared between DDCs. Residuary code is left in the main template. Includes of additional templates (<sapi:include ..>), DDC (<sapi:apply ..>) and content queries, variable, expressions (<sapi:apply .. >) are put to appropriate places.

Functionality Script Creation

Functionality scripts (DDCs) are described in XML Sapiens 2.0 specification. DDC files of SAPID CMF are located in folder /views/delivery/ddcs/ and file name necessarily must be the same with DDC name.

Output of enumeration variables is defined as &this.this.VARIABLE.value;, wherein the first this means its DDC and the second this – enumeration of the DDC. Thus variables of a nested enumeration should be named as &this.this.this.VARIABLE.value;.

Example:

```
<?xml version="1.0"?>
<sapi version="2.0" xmlns:sapi="http://www.xmlsapiens.org/spec/sapi.dtd">
  <sapi:ddc name="test" title="ddc" ns="site1" category="navigation">
    <sapi:choose>
      <sapi:when exp="1">
        <sapi:for-each select="get_infochannel()" name="enum" title="Get
channel">
          <sapi:params>
            <sapi:param name="sources">4v*:link</sapi:param>
            <sapi:param name="limit">0,10</sapi:param>
            <sapi:param name="name">test</sapi:param>
          </sapi:params>
          <sapi:ifempty>Records was not found</sapi:ifempty>
          <sapi:fallback>
            infochannel CMS-application error
          </sapi:fallback>
          <sapi:choose>
            <sapi:when exp="1">
              <sapi:code>
                <b><a
href="&this.this.href.value;">&this.this.name.value;</a>
&this.this.channel_id.value;</b><br />
              </sapi:code>
            </sapi:when>
          </sapi:choose>
        </sapi:for-each>
      </sapi:when>
    </sapi:choose>
  </sapi:ddc>
</sapi>
```

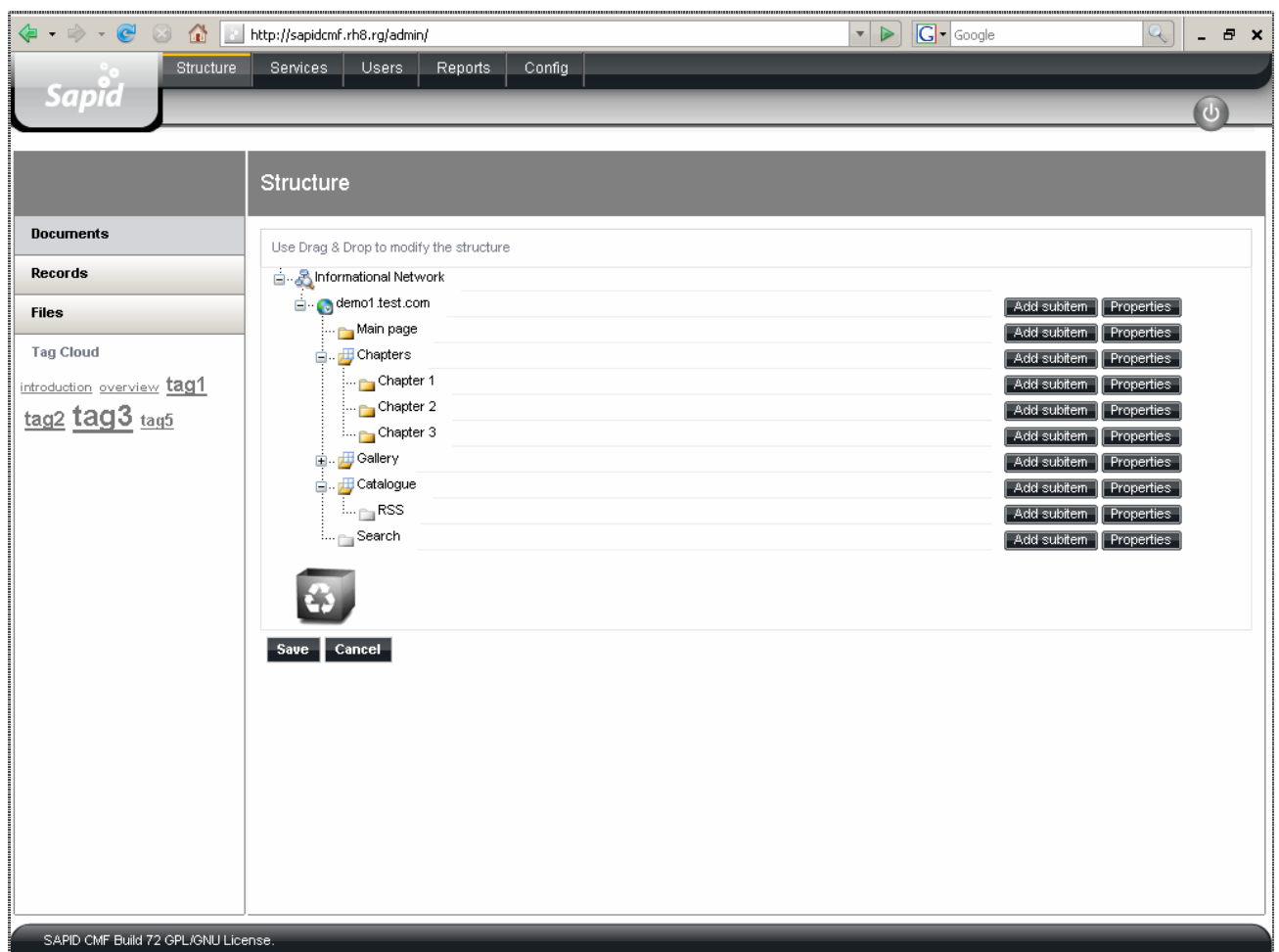
CMS-application are used for functionality marking to get data from the framework. See CMS-application section.

Web-project Administration

Administrative Panel

Address of the administrative panel you have denoted within framework installation (on default it's <http://yoursite/admin/>). When you type this address in your web-browser you will see a form to be requested of authorization data.

When you are authorized you can see the first application of the administrative interface (on default it's structure tree administrative panel)



Top horizontal menu of administrative panel has sections: Structure, Services, Users, Reports, Config. You can extend this list on your own.

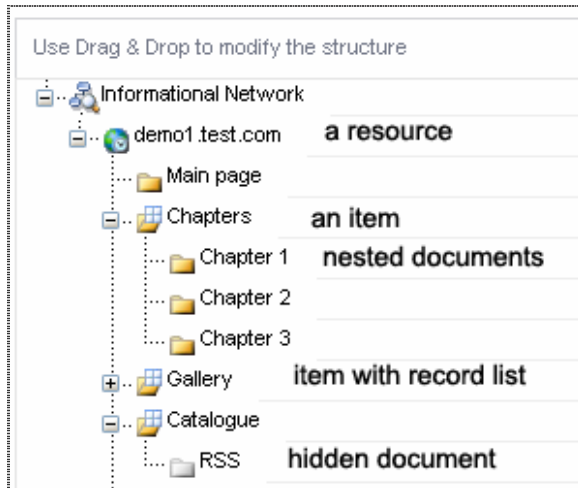
Left vertical menu contains list of applications of the section. Button “Log-out” is located in the top right corner.

Structure Section

This section is designated to manage documents, records, files of the web-project.

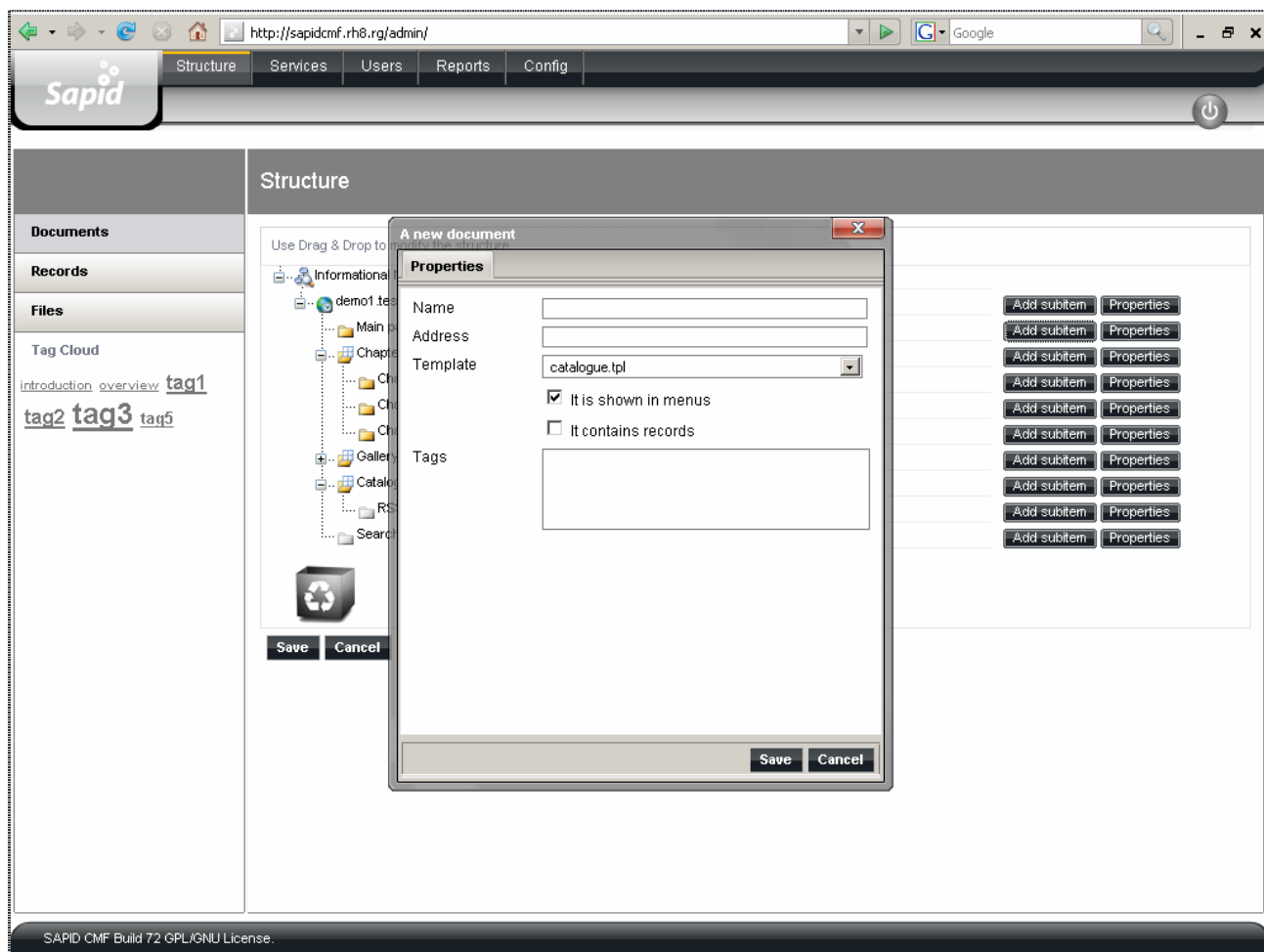
Document Management

Structure tree interface is well known to you. It's very similar such applications as Microsoft Windows Explorer. You can open/close tree branches with mouse. You can move or copy tree items with mouse (keep Ctrl pressed to copy a document/branch). You can put tree items into "recycle bin" to delete them.



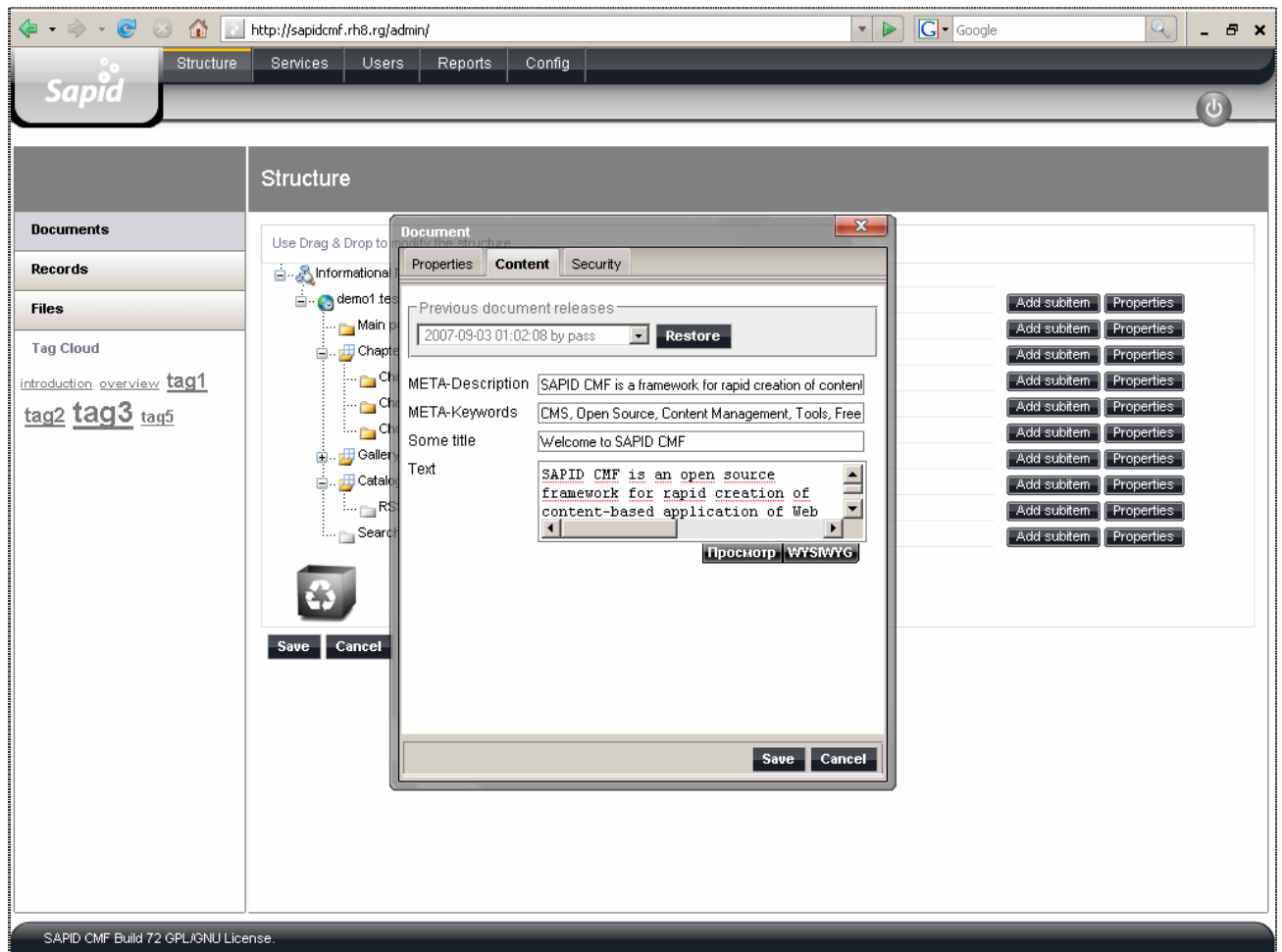
There is buttons "Add subitem" and "Properties" against each tree item

Press "Create subitem" against a tree item. You will see a window with some input fields. You will be requested for new document name, its address (it means only last part of address. For example: aboutus), its template file, its tags (keywords if you are going to associate documents between each other). Also it will be requested if you are planning to show the item in menus and if you want to add record list to the item).



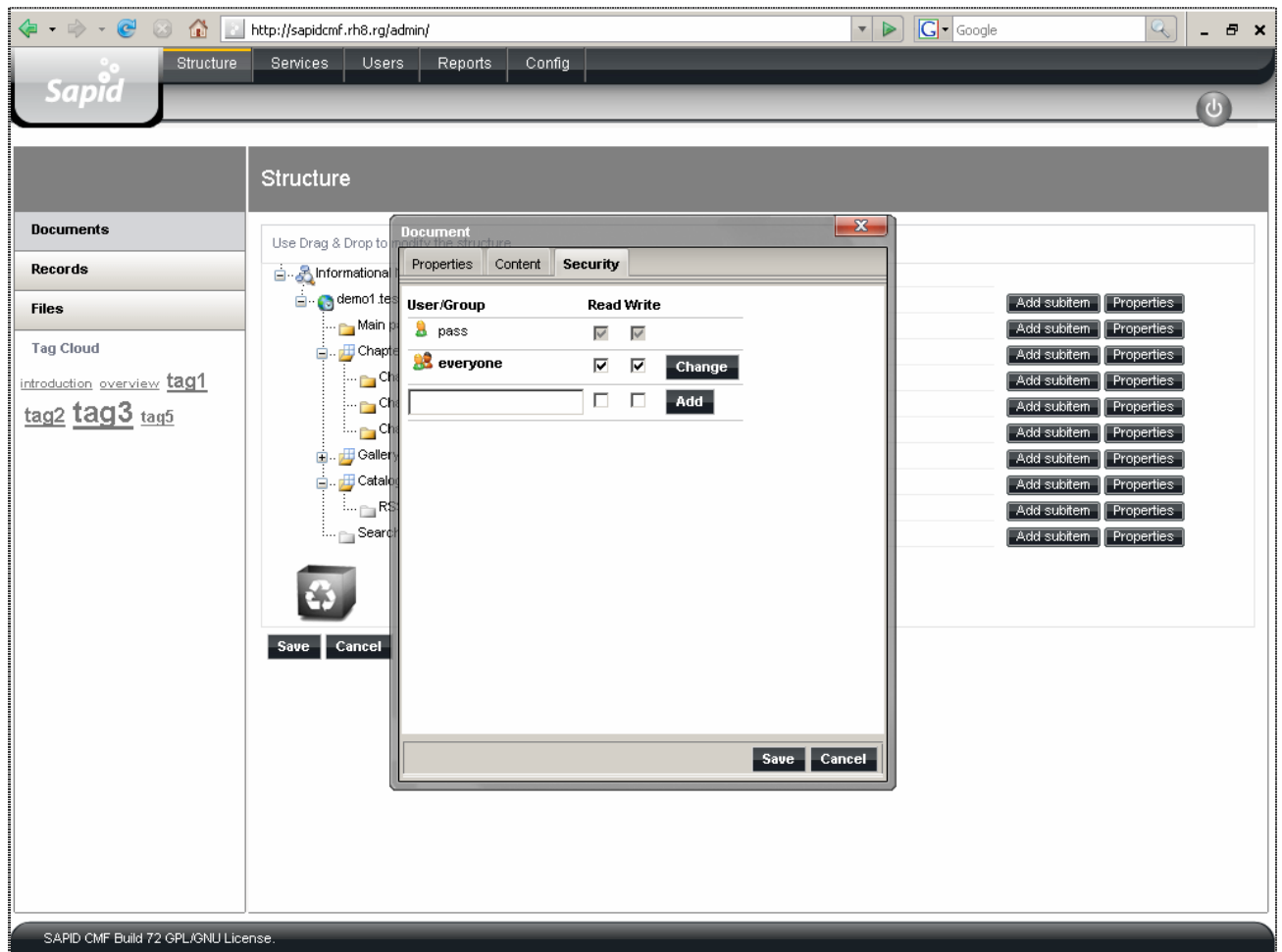
When you have filled all the fields and click “Save”, window will close and you can see new item in the structure.

If you now press “Properties” button of the item you will see a window with tabs “Properties”, “Content” and “Security”. Tab “Properties” has input fields to describe document attributes. We filled them in the previous example. Tab “Content” is formed based-on the template, assigned as template of the item in tab “Properties”. All QC (content queries) of the template will be presented there to request item content. Template files are located in folder `views/delivery/templates/`



You should know every time when you click Save in this tab, the framework saves entered content into version archive. So you will have as many content versions as you designated in configuration variables DOCCOPIESMAXNUMBER/ DOCCOPIESMAXNUMBERPERUSER. Here you can restore one of the previous versions as well.

Tab "Security" contains access list for web-project users and groups.



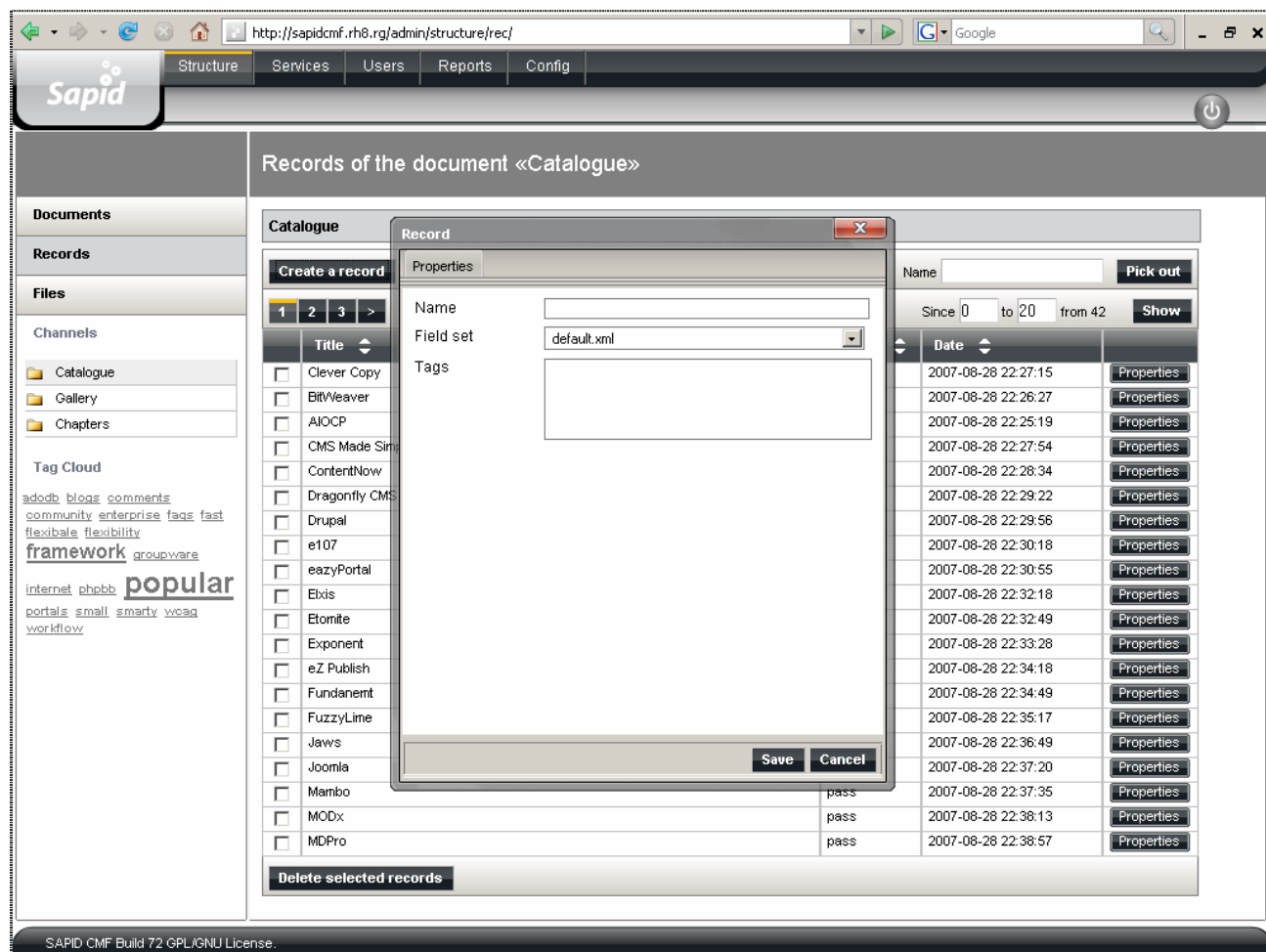
Record Management

Any document can contain record list. You just can mark checkbox "it contains records" in properties of the document and you will have in Records application of Structure section document name in the list of available record lists (beneath application list). Now you can choose this name in the list. You will get grid interface to manage records of the record list.

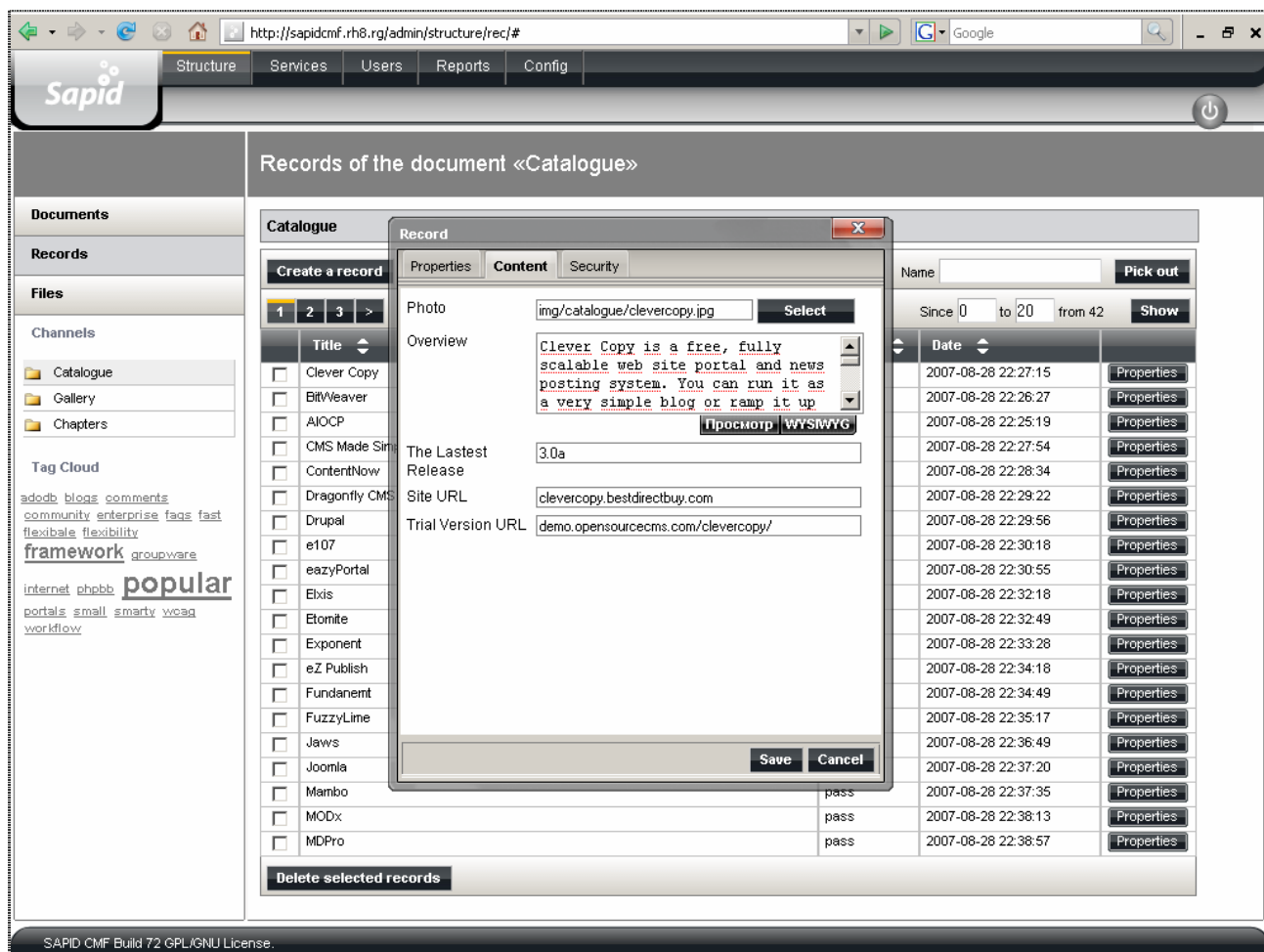
The screenshot shows the SAPID CMF administration interface. The browser address bar displays `http://sapdcmf.rh8.rg/admin/structure/rec/`. The interface has a top navigation bar with tabs: Structure, Services, Users, Reports, and Config. A left sidebar contains a 'Documents' section with 'Records' and 'Files' sub-sections. Under 'Files', there are 'Channels' (Catalogue, Gallery, Chapters) and a 'Tag Cloud' with various tags like 'adodb', 'blogs', 'comments', etc. The main content area is titled 'Records of the document «Catalogue»'. It features a 'Catalogue' header with 'Create a record' and 'Import' buttons, a search field, and a 'Pick out' button. Below this is a table with columns: Title, Owner, Date, and a 'Properties' button for each row. The table lists 20 records, all with 'pass' as the owner and dates from 2007-08-28. At the bottom of the table is a 'Delete selected records' button. The footer of the interface states 'SAPID CMF Build 72 GPL/GNU License.'

Title	Owner	Date	Properties
<input type="checkbox"/> Clever Copy	pass	2007-08-28 22:27:15	Properties
<input type="checkbox"/> BitWeaver	pass	2007-08-28 22:26:27	Properties
<input type="checkbox"/> AIOCP	pass	2007-08-28 22:25:19	Properties
<input type="checkbox"/> CMS Made Simple	pass	2007-08-28 22:27:54	Properties
<input type="checkbox"/> ContentNow	pass	2007-08-28 22:28:34	Properties
<input type="checkbox"/> Dragonfly CMS	pass	2007-08-28 22:29:22	Properties
<input type="checkbox"/> Drupal	pass	2007-08-28 22:29:56	Properties
<input type="checkbox"/> e107	pass	2007-08-28 22:30:18	Properties
<input type="checkbox"/> eazyPortal	pass	2007-08-28 22:30:55	Properties
<input type="checkbox"/> Elxis	pass	2007-08-28 22:32:18	Properties
<input type="checkbox"/> Etonite	pass	2007-08-28 22:32:49	Properties
<input type="checkbox"/> Exponent	pass	2007-08-28 22:33:28	Properties
<input type="checkbox"/> eZ Publish	pass	2007-08-28 22:34:18	Properties
<input type="checkbox"/> Fundanemt	pass	2007-08-28 22:34:49	Properties
<input type="checkbox"/> FuzzyLime	pass	2007-08-28 22:35:17	Properties
<input type="checkbox"/> Jaws	pass	2007-08-28 22:36:49	Properties
<input type="checkbox"/> Joomla	pass	2007-08-28 22:37:20	Properties
<input type="checkbox"/> Mambo	pass	2007-08-28 22:37:35	Properties
<input type="checkbox"/> MODx	pass	2007-08-28 22:38:13	Properties
<input type="checkbox"/> MDPro	pass	2007-08-28 22:38:57	Properties

Click "Create a record" and you see a window to request you new record properties.



You will be requested with name of the record, its field set filename and its tags. As in the case of document properties window tags are optional information which can be used to associate records between each other. When you fill the fields and press Save, a new record will appeared in the list. Now you can click Properties button against its name in the list. You will see a window with tabs: Properties, Content and Security. It reminds document Properties window, doesn't it? The first tab as you could see allows manage record attributes. The second one is meant to manage record content and its versions.



Content query fields of the tab are taken from record field set structure. Field set files are located in folder `views/delivery/fieldsets/`.

Tab "Security" manages user access list to the record.

As it was said a record can contain record list as well. If you want a record has its own record list, you need field sets for its nested records. Those field sets have to has pointer to parent record field set, like this:

```
<?xml version="1.0"?>
<sapi version="1.0" xmlns:sapi="http://www.xmlsapiens.org/spec/sapi.dtd">
<sapi:fieldset type="default" state="admin" title="">
  <sapi:apply name="qc.recontent.value" type="article"
title="Description" />
  <sapi:apply name="qc.rectable.value" type="article"
title="Specification" />
  <sapi:adopt fieldset="default" />
</sapi:fieldset>
</sapi>
```

The field set specified in `sapi:adopt` element is parent for records with that field set. So, for example, you have field set album (album.xml) for records which mean photo galleries and field set photo for nested records which mean photos of the galleries. Field set photo has line: `<sapi:adopt fieldset="album" />`. When you assign field set album to a record, you will see the record got a link. Click the link and you see record list of the record (photo list

of the gallery). Moreover when you will see in the list of available field set in Properties window only photo field set, because only this field set has album field set as a parent one.

File Management

Files application of Structure section has folders menu beneath application list. You can choose any folder of presented ones. You will see its files in grid. This list contains Upload button to add file in the list. You can see file content and delete files as well.

The screenshot shows the SAPID CMF Files management interface. The sidebar on the left has a 'Files' section with a folder tree. The main area displays a table of files with columns for Filename, Size, Date, and a 'View' button. A 'Delete selected records' button is at the bottom of the table.

Filename	Size	Date	View
album_sample1.jpg	48.3 KB	26.08.2007 16:18	View
c-bury.png	370 bytes	26.08.2007 19:34	View
c-vote.png	402 bytes	26.08.2007 19:34	View
close_btn.gif	85.0 bytes	10.08.2007 21:28	View
frame.gif	1.30 KB	26.08.2007 18:23	View
leftcor.gif	211 bytes	10.08.2007 21:28	View
livedemo.gif	7.83 KB	02.09.2007 14:28	View
loading.gif	2.70 KB	10.08.2007 21:28	View
openid.gif	550 bytes	03.09.2007 20:52	View
pframe.gif	1.19 KB	06.09.2007 22:10	View
rightcor.gif	212 bytes	10.08.2007 21:28	View
rss_icon.jpg	1.03 KB	06.09.2007 22:15	View
sample1.gif	327 bytes	10.08.2007 21:28	View
working.gif	1.12 KB	10.08.2007 21:28	View
x.gif	43.0 bytes	10.08.2007 21:28	View

Services Section

The section contains applications "Advertising", "Backup", "System console", "Maillist", "Thesaurus". Besides you can enrich the section with your own applications (See "Administrative Application Creation" section).

Advertising Management

You can find a solution to manage advertising on you web-project in distributive pack. How does it work? You just put into template or DDC following line to assign ad place:

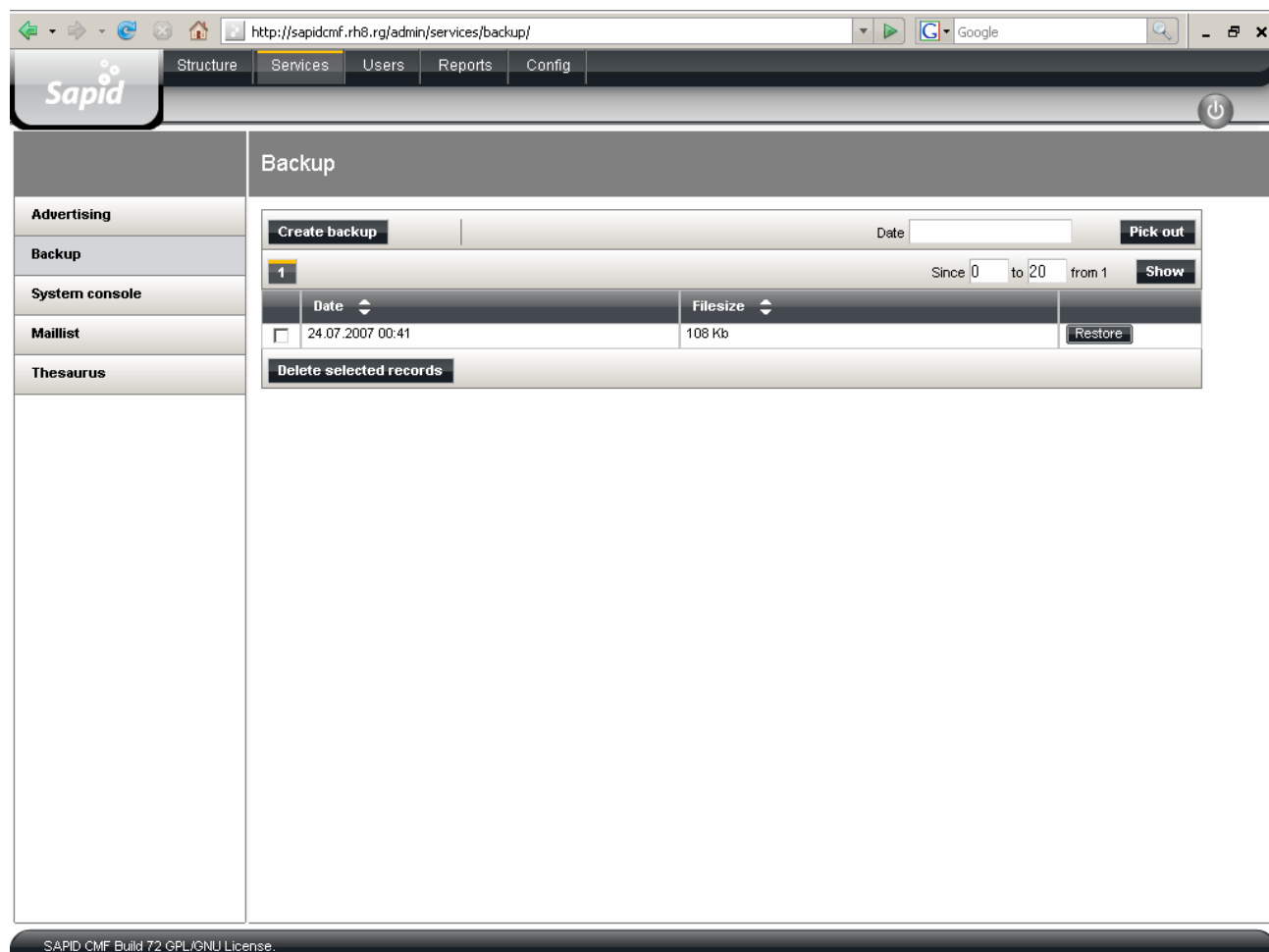
```
<sapi:apply name="ddc.ad.value" />
```

Ad which will be shown at the position is managed in Advertising application of Services section. You can add a new banner to be rotated. You will be requested for banner name, address to redirect, its HTML-code or graphical file. Banner statistics will be presented in the banner list in Advertising application. Naturally you can change banner's properties.

The screenshot displays the SAPID CMF Admin interface. The browser address bar shows <http://sapdcmf.rh8.rg/admin/services/#>. The interface has a sidebar on the left with the following links: Advertising, Backup, System console, Maillist, and Thesaurus. The main content area is titled 'Services' and contains a table with the following columns: Title, Showed, and Clicked. The table lists several services, including 'Site Guide Toolkit', 'AntiSpam Feedback Toolkit', 'Site Sapiens 3.0 ECMP', 'Thesaurus Tooltip Kit', and 'Easy Grid'. A 'Properties' dialog box is open over the 'Site Sapiens 3.0 ECMP' service, showing fields for Name, URL, HTML Code, and Banner. The 'Banner' field contains the value 'img/banners/sitesapiens.gif' and the 'Activate' checkbox is checked. The dialog box also has 'Save' and 'Cancel' buttons at the bottom.

Backup Management

You can make DB backups and you have Backup application of services section for it. The application has “Create backup” button. So you can use it to create a backup of DB. If you want to restore one of DB copies, click “Restore” button against version in the list.



System Console

The section serves to manage the framework on low level. You can assign one of console modes in field “Command”. The console support modes: API:, SITE:, SQL

API mode means direct input of framework API commands. For example, `print_r($doc->get(1));`

SQL mode is used to assign SQL commands, say `SELECT * FROM sf_thesaurus LIMIT 0,10`

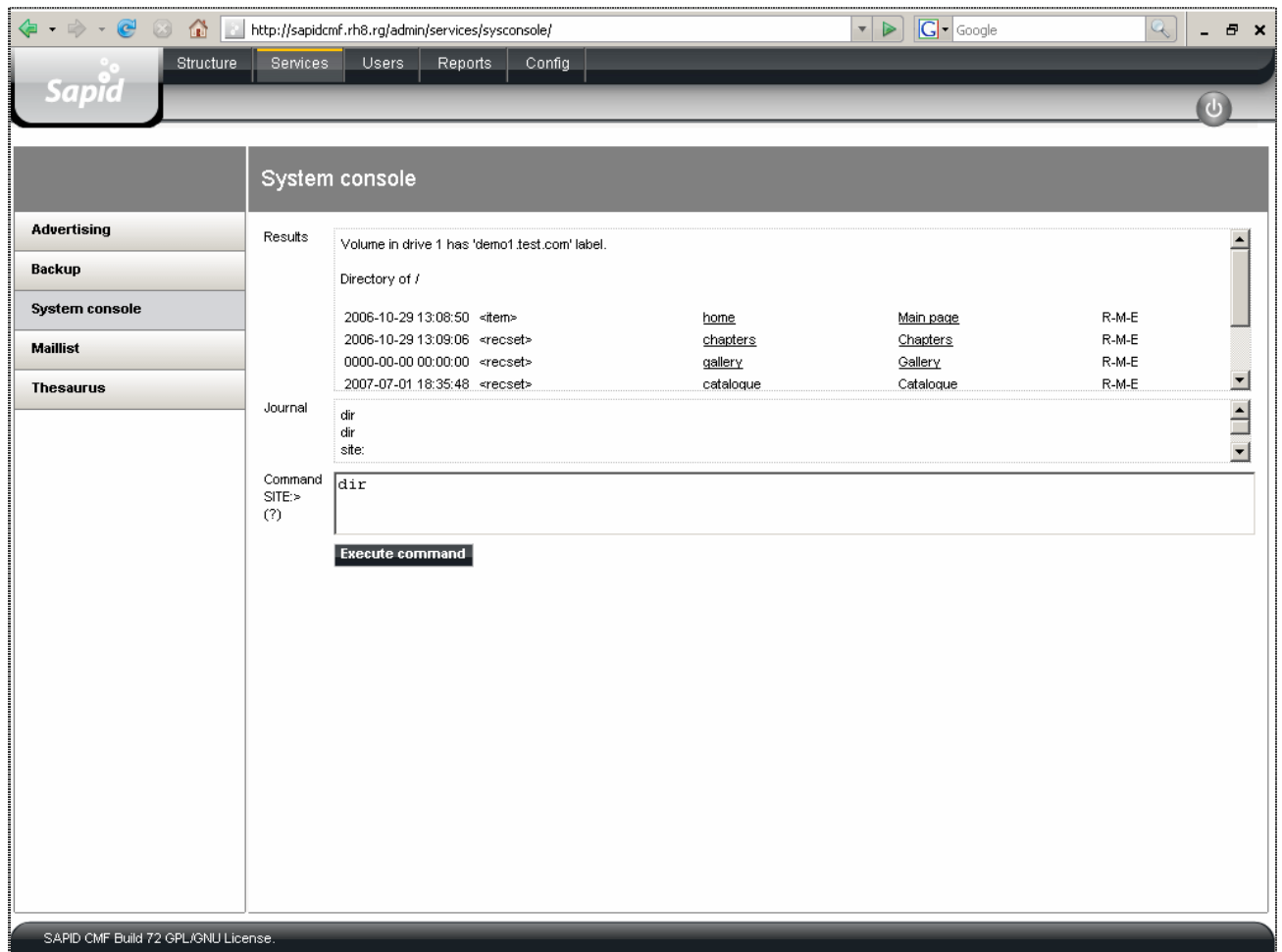
SITE mode can be used to manage structures and content. At the moment following commands are available:

Help – to get help on a command

Dir – to show item content

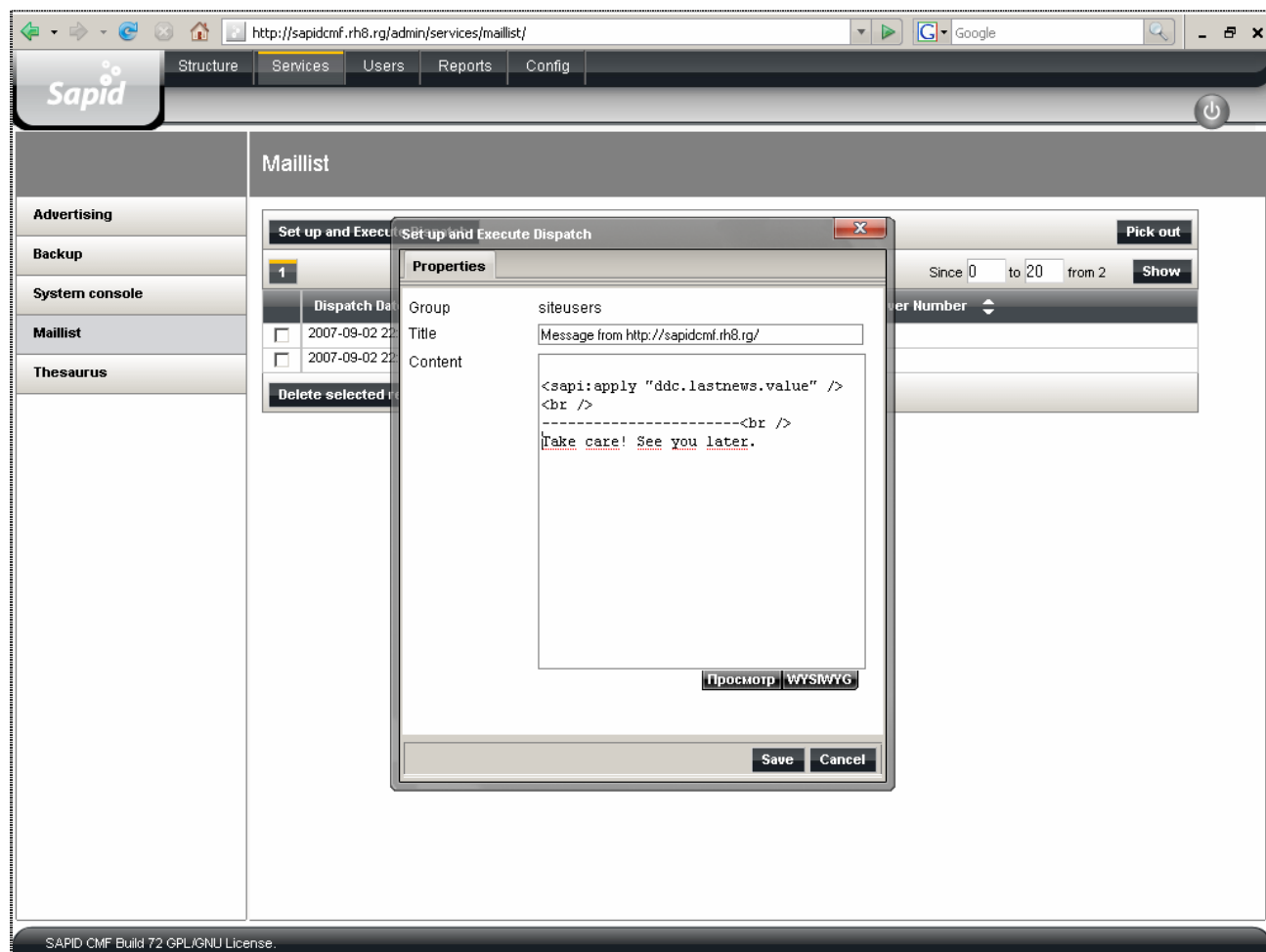
Cd – to choose item

Index – to reindex tags of documents and records.



Maillist Management

You can set up and execute dispatches on you web-project. "Maillist" application of "Services" section allows to do this. Click "Set up and execute dispatch" and enter email subject and body. You can use within the body XML Sapiens elements. So you can put there `sapi:include` or `sapid:apply` to have functionality script in the body.



Thesaurus

You can manage terms dictionary in Thesaurus application of the Services section.

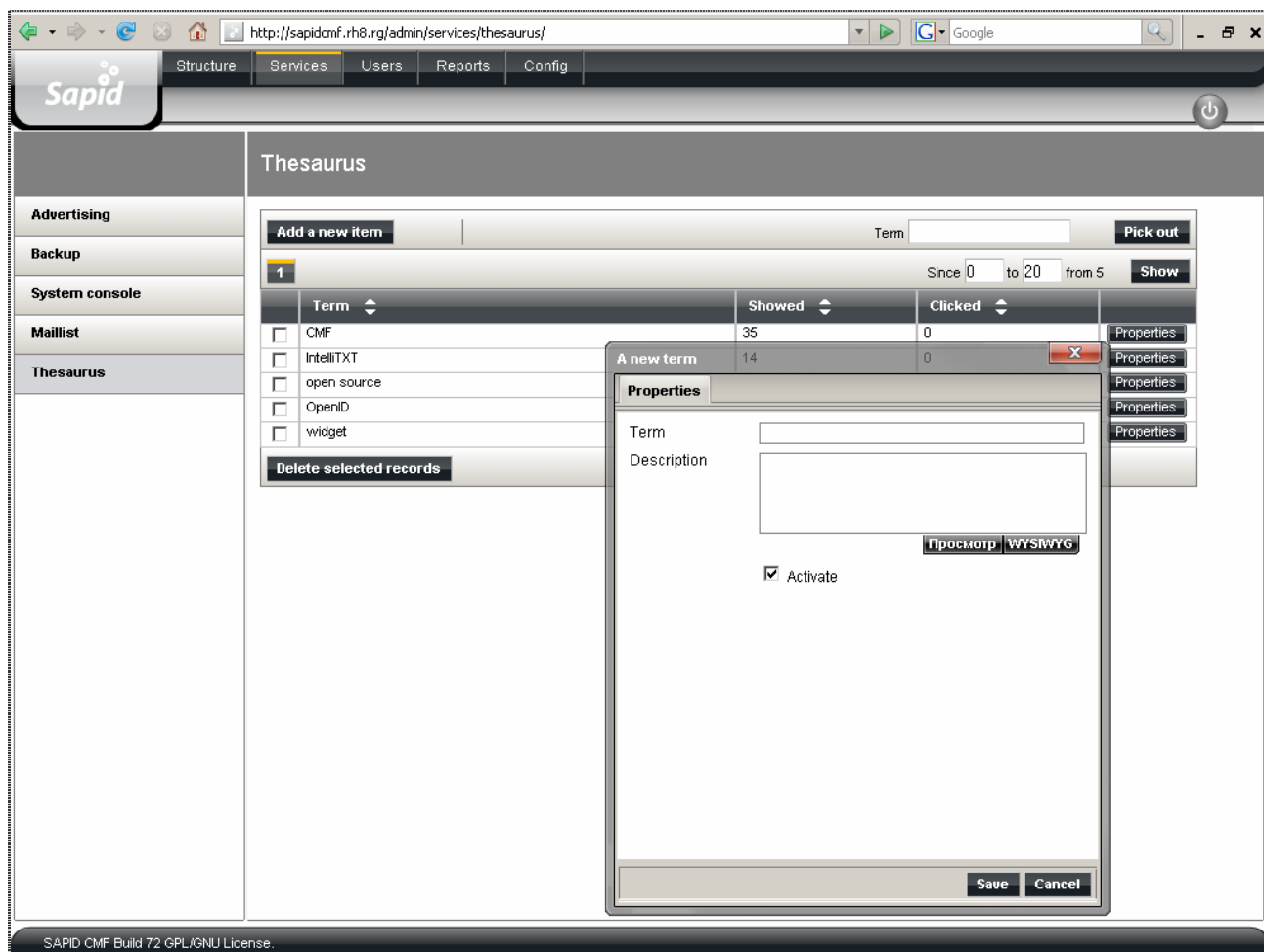
Use following code to get tooltips with term definitions on your page content:

```

<script src = "&http_path.value;vendors/yui/yahoo/yahoo.js" ></script>
<script src = "&http_path.value;vendors/yui/connection/connection-min.js"
></script>
<script src = "&http_path.value;views/delivery/js/thesaurus.js" ></script>
<script>
    THConfig.parseElements = ['PageContent', 'Footer'];
    <sapi:apply exp="isequal('&state.value;', 'edit', '///','')" />
    parseContent();
</script>

```

Now all terms on a page which are described in the thesaurus will be transformed into dotted links. If somebody hovers mouse cursor on a term there will be shown its description



Users

You can manage your list of users and groups on the framework by means “Management” application of the “User” section. It has grid with the list of all users and groups on the framework. However you can filter this list with “User Groups” menu.

Choose in “User Groups” list a group. Now click “Create a user” button to add a user in the group. You will be requested for new user login, password, full name, email, preferred interface language, profile field set filename, if user has to be activated and if he/she will have permission to create documents/records.

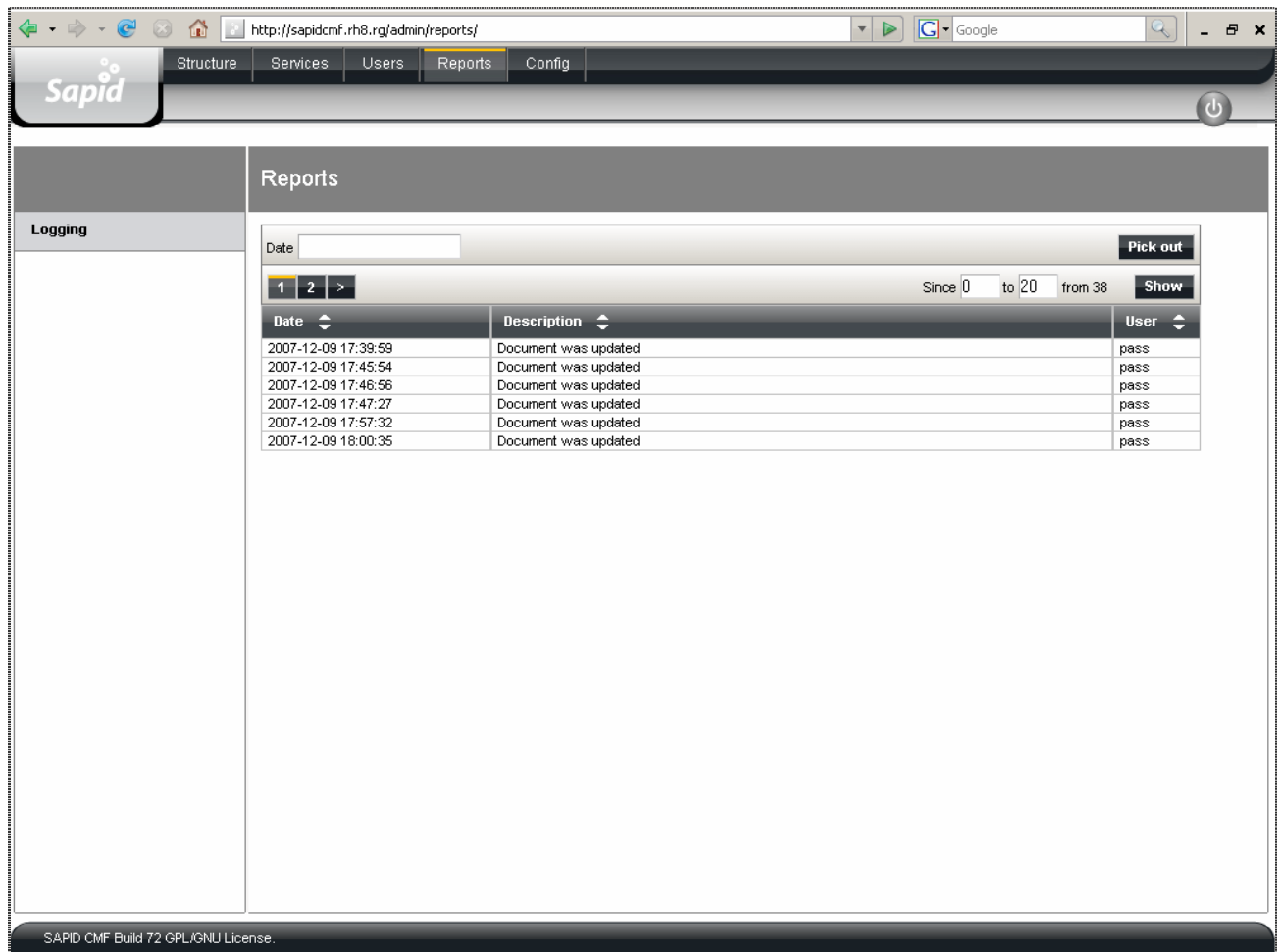
To have a new group in “User Group” list click “Create a group” button. Enter name and variable of the group.

The screenshot displays the SAPID CMF Users management interface. The browser address bar shows the URL `http://sapdcmf.rh8.rg/admin/users/`. The interface features a top navigation bar with tabs for 'Structure', 'Services', 'Users', 'Reports', and 'Config'. A left sidebar under the 'Management' section lists 'User groups' including 'All', 'Developers', 'Group A', 'Group B', and 'Site Users'. The main content area is titled 'Users' and contains a 'Create a user' dialog box, a 'User properties' dialog box, and a table of users. The 'User properties' dialog box is open, showing fields for Login, Password, Full name, Email, Language (set to 'en'), and Profile (set to 'default.tpl'). It also has checkboxes for 'Activate' and 'Creation permission'. The table on the right lists users with columns for 'Login date' and 'Properties'.

Login date	Properties
-24 19:34:53	Properties
-24 17:48:52	Properties
-24 17:48:35	Properties
-16 00:06:15	Properties
-24 21:00:55	Properties
-24 21:00:28	Properties
-24 17:43:47	Properties
-26 09:09:09	Properties
-15 23:08:47	Properties
-26 09:09:09	Properties
-24 19:37:36	Properties
-24 19:33:39	Properties
-26 09:09:09	Properties

Reports

“Report” section now contains only “Loggingin” report – monitor about what happens in the framework. But you can add your own report applications there.

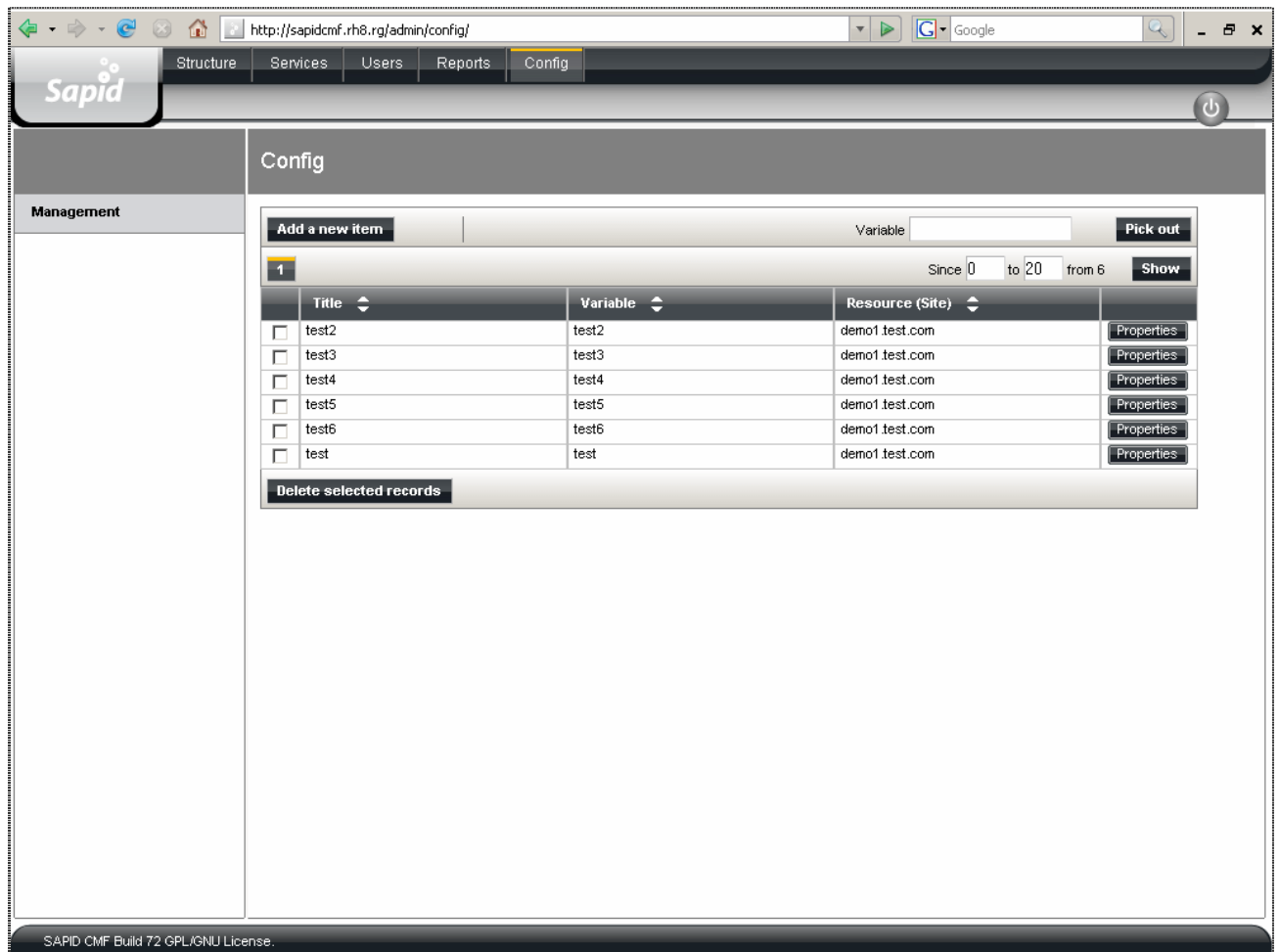


The screenshot shows the SAPID CMF web application interface. The browser address bar displays <http://sapdcmf.rh8.rg/admin/reports/>. The navigation bar includes tabs for Structure, Services, Users, Reports, and Config. The Reports section is active, showing a table of logs. The table has three columns: Date, Description, and User. The data shows five entries of 'Document was updated' by user 'pass' on 2007-12-09. A 'Logging' sidebar is visible on the left, and a footer at the bottom indicates 'SAPID CMF Build 72 GPL/GNU License.'

Date	Description	User
2007-12-09 17:39:59	Document was updated	pass
2007-12-09 17:45:54	Document was updated	pass
2007-12-09 17:46:56	Document was updated	pass
2007-12-09 17:47:27	Document was updated	pass
2007-12-09 17:57:32	Document was updated	pass
2007-12-09 18:00:35	Document was updated	pass

Config Section

“Management” application of “Config” section allows to define additional configuration variables and control their values. Those variables will be available as `config_variable.value`

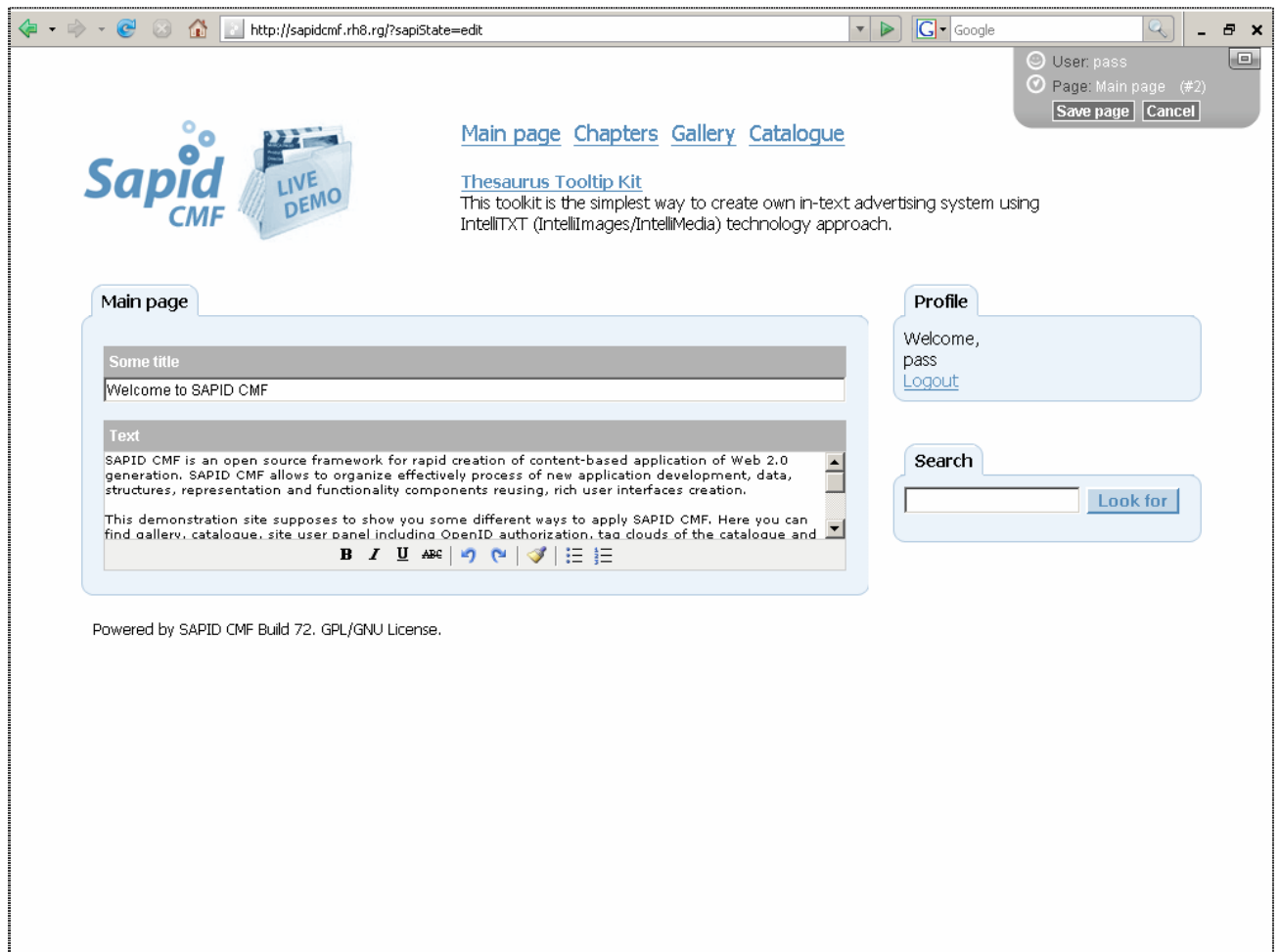


Inline Administrative Mode

You can control your site in INLINE administrative mode. If you are authorized as a user of developers group, you will see on site pages control panel.



The panel contains name of the authorized user, name and ID of the open page. "Edit page" button allows go for editing of the page. In that case you will instead of content input fields.



You should fill them and press "Save" button of the control panel.

If the page contains record list and DDC of the list has INLINE elements (See /views/delivery/ddcs/infochannel.xml sample) and user is authorized as administrator (a user of developers group), you will see above the list "Add" button and "Change" / "Delete" buttons against each record.

http://sapdcmf.rh8.rg/catalogue/

User: pass
Channel: Catalogue (#38)
Edit page Logout

[Main page](#) [Chapters](#) [Gallery](#) [Catalogue](#)

[Easy Grid Toolkit](#)
Easy Grid is a free toolkit to create your own manageable data-GRIDs using AJAX

Catalogue

[Главная](#) > [Catalogue](#)

Add

SAPID CMF [Change](#) [Delete](#)

Sapid
SAPID CMF is a free framework for rapid creation of content-based application of Web 2.0 generation. SAPID CMF allows to organize effectively process of new application development, data, structures, representation and functionality components reusing, rich user interfaces creation.

YACS [Change](#) [Delete](#)

YACS
Yet Another Community System (YACS) is an open source, open-standards content management system based on PHP and MySQL with a focus on code quality and best practices. The YACS product has a straightforward and modular component architecture that covers most common needs of demanding web masters. By design, YACS makes an efficient usage of computing and networking resources, and it allows for several kinds of extensions.

XOOPS [Change](#) [Delete](#)

XOOPS
XOOPS is a program that allows administrators to easily create dynamic websites with great content and many outstanding features. It is an ideal tool for developing small to large dynamic community.

Profile

Welcome,
pass
[Logout](#)

Tag Cloud

[adodb](#) [blogs](#) [comments](#) [community](#) [enterprise](#)
[faq](#) [fast](#) [flexible](#) [flexibility](#) [framework](#)
[groupware](#) [internet](#) [phpbb](#) [popular](#)
[portals](#) [small](#) [smarty](#) [wcaq](#) [workflow](#)

Widget

You can use this code to get catalogue/gallery RSS widget within your blog

```
<OBJECT  
classid="clsid:D27CDB6E-AE6D-11cf-96B8  
codebase="http://download.macromedia  
WIDTH="400" HEIGHT="400"  
id="RssReader" ALIGN=""> <PARAM  
NAME=movie  
VALUE="http://sapdcmf.rh8.rg/widgets/  
<PARAM NAME=quality VALUE=high>  
<PARAM NAME=bgcolor  
VALUE=#FFFFFF> <EMBED  
src="http://sapdcmf.rh8.rg/widgets/rssrs
```

If you click "Add" button you will see a window to be requested for new record properties like it was when you added records in "Records" application of the administrative panel. So you can fill input fields and press "Save" to see a new record in the list. Actually you need to fill content of record now. Click "Change" button against the record and fill content.

Project Debugging

You should assume 1 to DEBUG variable of the configuration file (config/rc.conf.php). In this case if an error happens the framework show you the message.

Anyway error messages are put in the report in the file TMP/error.log.


```
        if(!isset($obj["AppParams"]["name"])) return false;
        if($obj["AppParams"]["name"]!="test") return false;

        if(file_exists(ROOT_PATH."tmp/ddcs/test.tmp")) {
            $saved =
unserialize(file_get_contents(ROOT_PATH."tmp/ddcs/test.tmp"));
            $obj["AppParams"]    = $saved["AppParams"];
            $obj["DDCParams"]    = $saved["DDCParams"];
            $obj["EnumAttrs"]    = $saved["EnumAttrs"];
            $obj["Results"]      = $saved["Results"];
            return true;
        } else return false;
    }

    function putDDCCache($obj) {
        if(!isset($obj["AppParams"])) return false;
        if(!isset($obj["AppParams"]["name"])) return false;
        if($obj["AppParams"]["name"]!="test") return false;
        toLog(serialize($obj),"a","ddcs/test.tmp");
        return true;
    }

    $cache = new Aspect();
    $pc = $cache->pointcut("call *::*");
    $pc->_event("getDDCCache", "getDDCCache(\$obj);");
    $pc->_event("putDDCCache", "putDDCCache(\$obj);");
    $pc->destroy();
    Aspect::apply($cache);
```

Defined Events

The_program_is_started – the framework is started, main libraries are loaded;

All_libraries_are_loaded – the framework is started, all libraries are loaded;

Error403Found – 403 error happend;

Error404Found –404 error happend;

ModalWindowSchemasDefined – modal window schemes of the administrative panel is defined;

Programm Code of Adaption

You have environment variables in \$env structure and common functions within adaptation code.

\$env Structure

\$env→SiteID –ID of the resource on default;

\$env→Document – array with data of the open document, wherein

Template – template filename;

Data – array with document content;

ID – document ID;

PointerID – reflected document ID;

DataPointerID – document ID to reflect data;

Redirect – redirect address;

\$env→URL – information of user request in address line;

\$env→User – information of the authorized user, wherein

ID – user ID;

Groups – user groups;

Profile – user profile file name;

CreationPermission – user create permission;

SysAdminPermission – sysadmin permission;

ProfileID – user profile ID;

\$env→Values – environment variables, wherein

args_length – number of address line arguments (for example, in this case /arg1/arg2/ variable will return 2);

extraargs_length – number of address line optional arguments (for example, in this case /arg1/arg2/data/extrarg1/ variable will return 1);

DirectAccessArgs_length – number of arguments when record is requested (for example, in this case /arg1/arg2/0000192/0000193/ variable will return 2);

document_url – the address of current document (for example, gallery/);

args.0 – the first argument of address line (for example, gallery). Any next argument is available in variables like args.index_of_argument;

extraargs.0 – the first argument of optional address line (for example, something). Any next argument is available in variables like extraargs.index_of_argument;

argsstring – screened string of address line arguments (for example, gallery/);

argsstringwithoutslash – the same, but without slashes (for example, gallery);

admin_view_http_path – address of administrative panel templates root folder (for example, <http://sapidcmf.rh8.rg/views/default/>);

`delivery_view_http_path` – address of site templates root folder (for example, <http://sapidcmf.rh8.rg/views/delivery/>);

`http_path` – current site address (for example, <http://sapidcmf.rh8.rg/>);

`admin_http_path` – administrative panel address (for example, <http://sapidcmf.rh8.rg/admin/>);

`system_configuration_date` – framework configuration date (for example, 2007/08/22);

`system_version` – framework version (for example, Build 85);

`user_id` – authorized user identifier (for example, pass);

`site_id` – open site identifier (for example, 1);

`record_id` – requested record identifier. If record is not requested it returns 0;

`user_fullname` – full name of authorized user;

`user_login` – login of authorized user;

`user_profile` – description template of authorized user (for example, default.tpl);

`document_title` – title of the open document (for example, Gallery);

`document_template` – template of the open document (for example, gallery.tpl);

`document_id` – identifier of the open document (for example, 9);

`document_data_x` – content data, wherein x is name of a content field;

`record_data_x` – content record, wherein x is name of a content field.

Common Functions

`http_path($val)` – returns site address by SiteID;

`format2recordid($val)` – returns special format for record ID, appropriate to be used in address line;

`getMicrotime()` – return time in microseconds;

`sortByKey(&$array, $sortby, $order='asc', $type=SORT_NUMERIC)` – sorts array by key;

`lang($key)` – return key definition from interface content base;

`d($array)` – show array content within debugging;

`array2table($Arr)` – shows 2-dimensional array as a table;

`trace()` – shows all system calls before function was started within debugging;

`toLog($str)` – outputs data into log file;

`array_trim($src_array, $exclusions_array)` – cleans array by array of exceptions;

Schema_xxx variable contains array defining modal window tabs {"ctrl":"controller address", "title":"caption"}, ...

You ought to define scheme of the modal window in the following plugin:

Adaptation plugin sample:

Plugins/index.php

```
// Sample of administrative panel modal window customization
include(ROOT_PATH."plugins/ModalWindowCustomization.inc.php");
$ModalWindowCustomization = new Aspect();
$pc = $ModalWindowCustomization->pointcut("call *:*");
$pc->_event("ModalWindowSchemasDefined",
"customizeModalWindow(&\$obj);");
$pc->destroy();
Aspect::apply($ModalWindowCustomization);
```

Plugins/ ModalWindowCustomization.inc.php

```
<?
function customizeModalWindow($ModalWindowSchemas) {
$ModalWindowSchemas['ShowDocModalWindow'] = 'var Schema_ShowDocModalWindow
= [{"ctrl":"admin/structure/doc/docproperties.ctrl.php",
"title":"' .lang("Properties") . '"},
{"ctrl":"admin/structure/doc/doccontent.ctrl.php","title":"Something
new"},
{"ctrl":"admin/structure/doc/security.ctrl.php",
"title":"' .lang("Security") . '"}];';
}
?>
```

Creation of Administrative Panel Presentation Themes

Presentation theme is defined in file config/rc.conf.php:

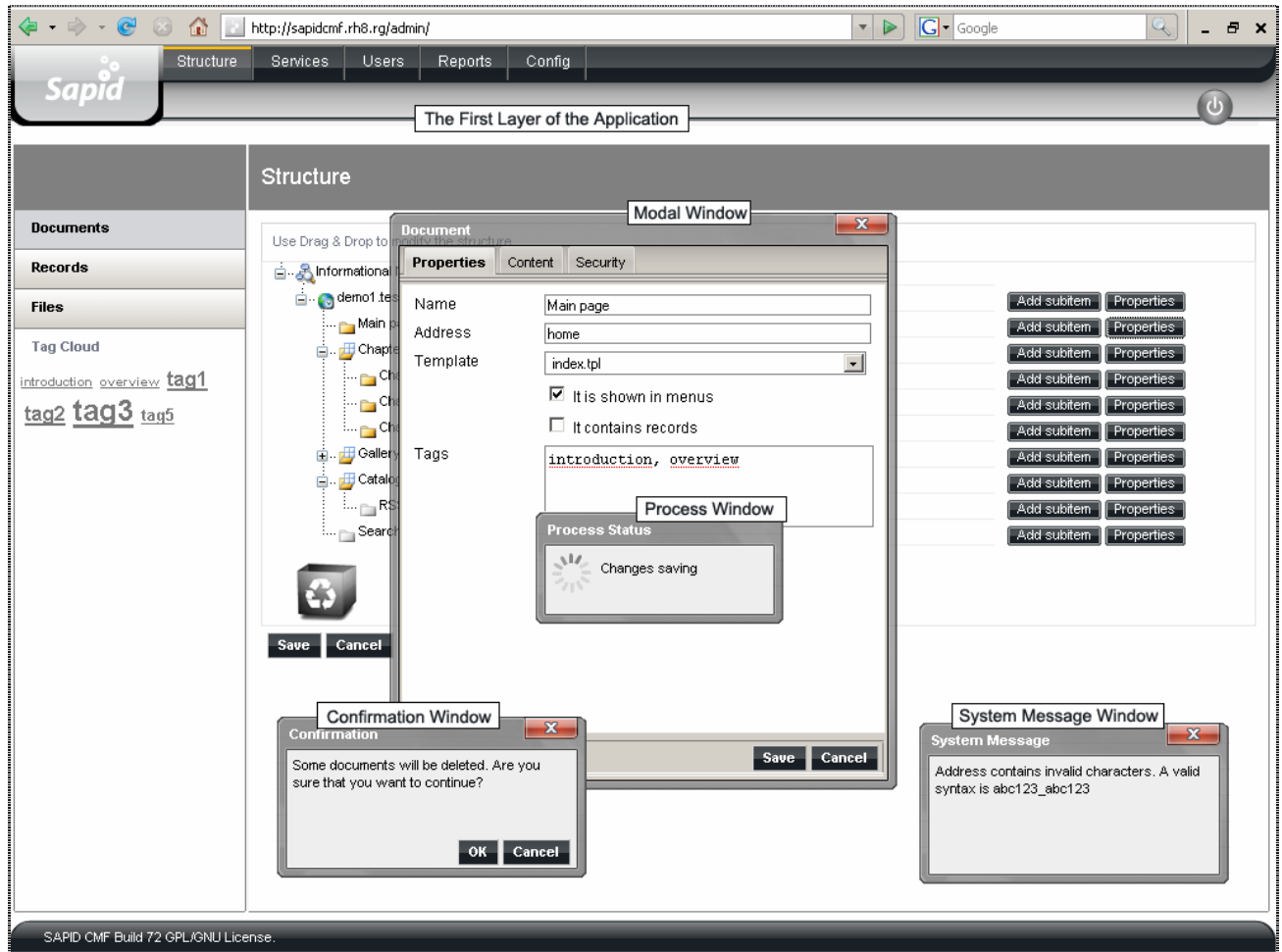
```
define("THEME", "default");
```

It's name if the folder, wherein theme templates and patterns are located

Administrative Application Creation

Administrative Application Structure

Any administrative application of the framework has the first layer, with horizontal menu of sections, vertical menu of applications and work desktop.



Elements of the first layer can be controlled by means modal window. If AJAX-controllers needs to report something into the administrative panel that message will be showed in system message window.

JS Syntax:

```
showSystemMessage("Text of the message");
```

When the framework is busy with some process it will be displayed by means process window.

JS Syntax:

```
showProcessMessage("Text of the message");  
hideProcessMessage();
```

There is a sort of windows with confirmation request. Those windows are used to request confirmation before operation will be executed.

JS Syntax:

```
showConfirmationMessage('Text of the message', okActionFunction);
```

How to Build Application

In order to build application, first of all you should define its section in horizontal menu of administrative panel. Say, if you want to add a new report you should choose section "Reports". Well, we open folder /app/admin/reports and find there application logs folder. We create our own folder by analogy. Let's name it ecommerce. Add there files index.layout.php, app.db, ecommerce.ctrl.php.

Access list in app.db can be following one:

```
<?xml version="1.0" encoding="utf-8" ?>
<treeitem titlekey="Ecommerce" sortkey="2">
    <acl>
        <group name="developers" permissions="7" />
    </acl>
</treeitem>
```

Wherein titlekey attribute is name of application (if language content base xx.lang.php doesn't contain translation for the name, application menu will have just a new item Ecommerce). Sortkey value points at application position in the menu.

Index.layout.php content can be like this:

```
<?php
$env->Values["window_interface_title"] = lang("Logging");
$tpl = factory("view", "template");
$output = $tpl->parseTemplate("#ecommerce.tpl");
?>
```

We assign application name to show it in work desktop and give control to the template of administrative panel: ecommerce.tpl

AS for ecommerce.ctrl.php controller, its content should be built on following scheme:

```
<?PHP
class Controller {
    var $json

    function Controller() {
        global $json;
        $this->jjson = &$json;
        return $this;
    }

    function remoteProcedure1() {
        //...
```

```
$this->json->Body = "...";  
$this->json->ErrorMsg = "...";  
$this->json->ActionCode = 1;  
}  
}  
$ctrl = new Controller();  
if(isset($_POST["call"]))  
    $ctrl->$_POST["call"]();  
?>
```

Request to the controller, containing in call POST variable name of remote procedure (for example, remoteProcedure1) initiates Controller class and execute requested procedure. Server response can be defined in the procedure. If procedure defines value to the \$this->json->Body variable this value will be showed in message window. Value of \$this->json->ErrorMsg is shown in system message window as a error. And value of \$this->json->ActionCode can be used to define what JS has to do after controller response.

Now let's create the template which is linked at index.layout.php. So, we create file /views/default/templates/ecommerce.tpl.

```
<sapi:include href="#subtemplates/header.tpl" />  
<link rel="STYLESHEET" type="text/css" href="/views/default/css/grid.css" />  
<script src="/views/default/js/json_grid.js" type="text/javascript"></script>  
<body onload="initInterface()">  
<sapi:include href="#subtemplates/toppanel.tpl" />  
    <div class="smf_caption">&nbsp;</div>  
    <sapi:apply name="ddc.#submenu.value">  
        <sapi:param name="root">&args.0.value;</sapi:param>  
    </sapi:apply>  
    <div class="smf_docinfo">  
    </div>  
<sapi:include href="#subtemplates/middle.tpl"/>  
  
    <div class="smf_place">  
        Work desktop code  
    </div>  
<sapi:include href="#subtemplates/footer.tpl"/>
```

A form, tree or grid can be presented in the place of work desktop code. In the case of a form we add input fields and buttons. Then we define events to the buttons and describe them in JS. For example,

```
function ourEventAction(param1, param2) {  
    showProcessMessage(lang("Please wait"));  
    serverRequest("admin/reports/ecommerce/ecommerce.ctrl.php", "call==remoteProcedure1&param1="+param1+"&param2="+param2, processDataRestoring, false);  
}
```

Now you can put in the desktop place a button like `<button onclick=" ourEventAction(1,1)">Execute</button>`

As for tree it's more difficult, but you can get to know with that stuff on sample in the template tree.tpl.

Creation of Applications with Lists

In the case of grid, we can put in the desktop place following code:

```
<sapi:apply name="ddc.#gridnav.value">
    <sapi:param name="grid_id">ecommerce</sapi:param>
    <sapi:param name="filters">
        <sapi:apply exp="lang('Date')" /> <input class="filter"
type="text" name="edate" value="" />
    </sapi:param>
</sapi:apply>

<script>
    EGConfig.sUrl = "admin/reports/ecommerce/ecommerce.ctrl.php";
    settings(columns, '{"field":"edate","title":lang("Date"),
"width":"25%"}');
    settings(columns, '{"field":"edesc","title":lang("Transaction"),
"width":"70%"}');
    settings(columns, '{"field":"user","title":lang("User"),
"width":"15%"}');
    makeRequest('');
</script>
```

Filters parameter of the DDC gridnav assign additional grid filters. Grid configuration is described in the JS:

EGConfig.sUrl – grid controller location;

Settings – list structure, wherein each column is described in the following way:

```
settings(columns, '{"field":"field name","title":lang("Field title"), "width":"List column width"}');
```

makeRequest("") command requests the server for the list.

In the sample the grid will contain 3 column and its content will be requested from ecommerce.ctrl.php controller. However the controller can't return list content yet. You should teach it. Let's rewrite controller that it can better serve our goals.

```
<?PHP
registerLib("view", "grid");

class Controller extends Grid {
    var $rec;
    function Controller() {
        $this->init();
    }
}
```



```
$this->rec = factory("model","model");
return $this;
}
function getData() {
    global $db;

    // Restore the last state of the grid
    if(!isset($_POST["call"]) AND !isset($_POST["offset"]) AND
!isset($_POST["limit"]) AND !isset($_POST["filter_field"])) {
        $settings = $this->rec->getUserSettings("ecommerce");
        if($settings) list($this->Offset, $this->Limit, $this-
>Filter) = $settings;
    }

    // Get to know number of record of ecommerce report table
    $info = $db->one("SELECT count(*) as counter FROM
ecommerce");
    $this->Length = $info["counter"];
    if($this->Offset<0) $this->Offset=0;
    if($this->Offset>$this->Length) $this->Offset=0;
    if($this->Limit>$this->Length) $this->Limit = $this->Length;
    // Apply filters if user has demanded this
    $filter_str = "";
    if($this->Filter) $filter_str = " WHERE ".$this-
>Filter["field"]." LIKE '{ $this->Filter["value"] }%' ";
    // Build SQL to get data for our report
    $sql = "SELECT * FROM ".LOGGINGTABLE." ".$filter_str."
".($this->OrderBy?" ORDER by ".$this->OrderBy["field"]." ".$this-
>OrderBy["direction"].":")." LIMIT { $this->Offset }, { $this->Limit }";
    $res = $db->all($sql);
    $key = 0;
    foreach ($res as $line) {
        foreach($this->Fields as $Index => $Field) {
            if($Field=="edesc") $line[$Field] =
lang($line[$Field]);
            $this->GridContent[$key][$Field]=$line[$Field];
        }
        $key++;
    }

    // Save this state of the grid
    $this->rec->putUserSettings("ecommerce", array($this->Offset,
$this->Limit, $this->Filter));
}
}

$ctrl = new Controller();
$ctrl->getData();
$json->Body = $ctrl->generateRespond();
?>
```

Now we can see in the application a grid with data of ecommerce table.

How to Make Lists Manageable

We saw how to build a report application in the previous example. You saw the grid of that application doesn't include any control elements to add, change or delete list records. So let's try to build a maillist control application, where we will need buttons to manage list records.

As in the previous case we create in the "Services" section(/app/admin/services/) folder maillist and locate files index.layout.php, app.db, maillist.ctrl.php there.

Application template (maillist.tpl) will be located in the folder /views/default/templates/. It's content will be following:

```
<sapi:include href="#subtemplates/header.tpl" />
<link rel="STYLESHEET" type="text/css" href="/views/default/css/grid.css" />
<script src="/views/default/js/json_grid.js" type="text/javascript"></script>

<script type="text/javascript">

var Schema_ShowItemModalWindow =
[{"ctrl":"admin/services/maillist/properties.model.php","title":"Свойства"}];

// Show modal window with document properties
// -- You should not change this function name. It's used in json_grid.js --
function showItemModalWindow(id){
    ModalWindowSaveAfterAction = "refresh grid & close";
    ModalWindowDataAssignment = "updateRecord";
    showModalWindow("admin/services/maillist/properties.model.php",
"Свойства", id, "admin/services/maillist/maillist.ctrl.php",
Schema_ShowItemModalWindow);
    return false;
}

// Show new modal window with document properties
function showNewItemModalWindow(parent_id){
    ModalWindowSaveAfterAction = "refresh grid & close";
    ModalWindowDataAssignment = "createRecord";
    showModalWindow("admin/services/maillist/properties.model.php",
"Добавить рассылку", parent_id, "admin/services/maillist/maillist.ctrl.php",
Schema_ShowItemModalWindow);
    return false;
}

// Processing when system gets data with AJAX
var processModalWindowContent= function(obj, empty) {
    if(respondStructure = responseAnalyze(obj)) {
        $("modal_window_body").innerHTML = respondStructure.Body;
    }
}
```

```
</script>

<body onload="initInterface()">
<sapi:include href="#subtemplates/toppanel.tpl" />

    <div class="smf_caption">&nbsp;</div>
    <sapi:apply name="ddc.#submenu.value">
        <sapi:param name="root">&args.0.value;</sapi:param>
    </sapi:apply>
    <div class="smf_docinfo">
    </div>

<sapi:include href="#subtemplates/middle.tpl"/>

    <div class="smf_place">

<!-- Tool panel -->

<sapi:apply name="ddc.#gridnav.value">
    <sapi:param name="grid_id">maillist</sapi:param>
    <sapi:param name="filters">
        <input class="smf_btn" type="button" name="add"
onclick="showNewItemModalWindow(false)" value="<sapi:apply exp="Assign
dispatch" />" />&nbsp;</sapi:param>
</sapi:apply>

<script>
    EGConfig.sUrl = "admin/services/maillist/maillist.ctrl.php";
    EGConfig.ContextMenu = "on";
    EGConfig.ContextMovingMenu="on";
    settings(columns, '{"field":"sdate","title":"Dispath date",
"width":"60%"}');
    settings(columns, '{"field":"receivers","title":"Number of receivers",
"width":"40%"}');
    makeRequest('');
</script>

    </div>

<sapi:include href="#subtemplates/footer.tpl"/>
```

As we see, here are Schema_ShowItemModalWindow JS structure and functions showItemModalWindow(), showNewItemModalWindow() и processModalWindowContent().

Schema_ShowItemModalWindow contains controller address and title of record properties control button. In this case record properties form is formed by the controller properties.model.php. You can see its content in the file /app/admin/services/maillist/properties.model.php:

```
<?
    $env->Document->Data["itemattr_fieldname"] = 'value;
    $output = parseXMLSapiens('
    <div class="smf_qc_area">
    <sapi:include href="#mlproperties.xml" parse="fieldset"
state="default" /></div>');
    $json->ActionCode = 1;
    $json->ErrorMsg = "";
    $json->Body = '""'.preg_replace("/[\r\n]/", "",
addslashes($output)).'';
?>
```

Input field values are put into \$env->Document->Data array. Then field set mlproperties.xml is given to XML Sapiens processor to be analyzed. The result is returned as content of model window. Content of field set can following:

```
<?xml version="1.0"?>
<sapi version="2.0" xmlns:sapi="http://www.xmlsapiens.org/spec/sapi.dtd">
<sapi:fieldset state="default" title="">
    <sapi:apply name="qc.itemattr_name.value" type="#inputtext"
title="Name" />
...
</sapi:fieldset>
</sapi>
```

Here is just enumeration of all record property fields. They will be filled with values of \$env->Document->Data.

showItemModalWindow() function defines, what will happen when “Properties” button against a record will be clicked. In this case there will be open modal window and properties.model.php controller will return the form into it. How the form will be formed we just said about. Window controller will be given record ID and name of the controller (maillist.ctrl.php) to be related. How? There was said in ModalWindowSaveAfterAction the grid will be refreshed and modal window closed after user clicks on “Save” button. So first properties.model.php controller will work, then related maillist.ctrl.php controller.

showNewItemModalWindow() function describes the modal window of “Assign dispatch” (“Add a record” in common case) by analogy.

In this case filter parameter of the DDC gridnav has button to assign dispatch (button to add a record in the list in common case). showNewItemModalWindow() function is defined to the button. We has already written this function.

Grid with manageable lists can have additional parameters:

EGConfig.ContextMenu = "on"; - context menu of the list is on (menu by right button of the mouse)

EGConfig.ContextMovingMenu="on"; - possibility to change record porions is available

Hmm, now we have got a list , but it still hasn't properties buttons against records. It's because we used grid controller of previous case. Let's change enumeration (foreach section) of method getData.

```
foreach($this->Fields as $Index => $Field) {  
    if($Field=="actions") $line[$Field] =  
        '<div style="margin-bottom: 3px;"><span class="grid_as">&nbsp;</span><a  
href="#" class="grid_action"  
onclick="showItemModalWindow('.$line["id"].')">'.lang('Properties').'</a><span  
class="grid_af">&nbsp;</span></div>';  
    $this->GridContent[$key][$Field]=$line[$Field];  
}
```

Wow! Now we have "Assign dispatch" button and properties buttons, but where is record delete button?

Let's add into list configuration in maillist.tpl the following line:

```
EGConfig.DeleteButton = true;
```

We should add in maillist.ctrl.php controller methods to delete record and change their positions.

```
function liftRecord() {  
    $info = $this->db->one("SELECT sort_id FROM ".MAILLISTTABLE."  
WHERE id={$_POST["id"]}");  
    $prev_info = $this->db->one("SELECT sort_id, id FROM  
".MAILLISTTABLE." WHERE sort_id>{$info["sort_id"]} ORDER by sort_id ASC");  
    $this->db->StartTransaction();  
    $this->db->update("UPDATE ".MAILLISTTABLE." SET  
sort_id={$prev_info["sort_id"]} WHERE id={$_POST["id"]}");  
    $this->db->update("UPDATE ".MAILLISTTABLE." SET  
sort_id={$info["sort_id"]} WHERE id={$prev_info["id"]}");  
    $this->db->CompleteTransaction();  
    return true;  
}  
  
function lowerRecord() {  
    $info = $this->db->one("SELECT sort_id FROM ".MAILLISTTABLE."  
WHERE id={$_POST["id"]}");  
    $next_info = $this->db->one("SELECT sort_id, id FROM  
".MAILLISTTABLE." WHERE sort_id<{$info["sort_id"]} ORDER by sort_id  
DESC");  
    $this->db->StartTransaction();  
    $this->db->update("UPDATE ".MAILLISTTABLE." SET  
sort_id={$next_info["sort_id"]} WHERE id={$_POST["id"]}");  
    $this->db->update("UPDATE ".MAILLISTTABLE." SET  
sort_id={$info["sort_id"]} WHERE id={$next_info["id"]}");  
    $this->db->CompleteTransaction();  
    return true;  
}
```

```
function deleteSelected() {  
    $this->db->StartTransaction();  
    foreach ($_POST as $field => $value) {  
        if(preg_match("/^EGRowDelete_/is", $field)) {  
            $rec_id = preg_replace("/^EGRowDelete_/is", "",  
$field);  
            $this->db->update("DELETE FROM ".MAILLISTTABLE."  
WHERE id={$rec_id}");  
        }  
    }  
    $this->db->CompleteTransaction();  
    return true;  
}
```

Ok. We have work application with manageable lists.

visits_counter	Visit counter
rate	Document rate
branchlength	Nested documents number
recsetslength	Nested records number
is_shown	Visibility flag

addSite()

Description: create a new site account.

Syntax: addSite(sting SiteName)

Return value: ID of created site.

```
$env->SiteID = $doc->addSite("Sample site");
```

add()

Description: create a new document account.

Syntax: add (array Attributs)

Return value: created document ID.

```
$PostData = array("name"=>"Item Example 1", "var"=>"item1",  
"parent_id"=>$env->SiteID);  
$id1 = $doc->add($PostData);
```

update()

Description: change document account data.

Syntax: update(ID, array Attributs)

Return value: true/false.

```
$PostData = array("name" => "New", "everyone"=>0);  
$doc->update(124, $PostData);
```

delete()

Description: delete document.

Syntax: delete(int ID)

Return value: true/false.

```
$doc->delete(102);
```

updateData()

Description: update document content.

Syntax: `updateData(ID, array Data)`

Return value: true/false.

Input parameters: ID, array "QC_name"=>"value"...

```
$PostData = array("title"=>"new", "message"=>"text");
$doc->updateData(124, $PostData);
```

copy()

Description: copy document to a new position in the structure tree.

Syntax: copy(int ID, int TargetParentID)

Return value: true/false.

```
$doc->copy(102, 43);
```


Return value: data array.

```
$res = $doc->BackupList(102);
```

RM API (Record Management Interface)

First of all you should initialize the class of the interface:

```
$rec = new Rec($env);
```

Record Definition

SAPID CMF record is an informational object, which can be presented in a linear list. Records are described with attributes, content, access list. Record field set is defined by an XML-file.

Document can include record set. Each record of the record set can include another record set.

Record Attributes

rec_id	Идентификатор
name	Заголовок
type	Тип записи (record или recset)
level	Record hierarchical level
site_id	Site ID
cdate	Record creation date
mdate	Record modification date
fieldset	Field set file name
sort_id	Sorting flag
owner	Owner ID
everyone	Permissions for everyone


```
$PostData = array("entry_name"=>"Modified Entry 1");
$rec->update(124, $PostData);
```

delete()

Description: delete record.

Syntax: delete(int ID)

Return value: true/false.

```
$rec->delete(1020);
```

updateData()

Description: change record data.

Syntax: `updateData(ID, array Data)`

Return value: true/false

Input parameters: ID, array "QC_name"=>"value"...

```
$PostData = array("title"=>"Title", "content"=>"Text");
$res = $rec->updateData(124, $PostData);
```

get()

Description: returns record attributes and content.

Syntax: get(int ID, constant WITHOUTDATA/WITHDATA)

Return value: array.

```
$data = $rec->get(1020, WITHOUTDATA);
```

getList()

Description: returns record list.

Синтаксис: `getList()`

See CMS-application `get_infochannel()`

Return value: data array.

getByCondition()

Description: returns record attributes and content by a condition.

Syntax: getByCondition(array/string Condition, constant WITHOUTDATA/WITHDATA)

Return value: array.

```
$data = $rec->getByCondition(array("url", "folder1/folder2/"),  
WITHOUTDATA);
```

getData()

Description: returns record content.

Syntax: get(int ID)

Return value: array

```
$data = $rec->getData(1020, WITHOUTDATA);
```

copy()

Description: copy record to a new position in the structure.

Syntax: copy(int ID, int TargetParentID)

Return value: true/false.

```
$rec->copy(1020, 34);
```

copyList()

Description: copy record list to a new position in the structure.

Syntax: copyList(int ParentID, int TargetParentID)

Return value: true/false.

```
$rec->copyList(33, 34);
```

addIndex()

Description: add an index to the record table.

Syntax: addIndex(sting IndexFieldName, int ParentID, [string DBFieldType])

Return value: true/false.

```
$rec->addIndex("price", 33, "VARCHAR( 32 )");
```

deleteIndex()

Description: delete additional index from DB table.

Syntax: deleteIndex(sting IndexFieldName)

Return value: true/false.

```
$rec->deleteIndex("price");
```



```
$attrs = $uapi->get ("sheiko");
```

update ()

Description: change attributes of user or group.

Syntax: update (string User, array Data)

Return value: true/false.

```
$uapi->update ("sheiko", array ("login"=>"newlogin"));
```

delete ()

Description: delete user or group.

Syntax: delete(string User)

Return value: true/false.

```
$uapi->delete ("sheiko");
```

getData ()

Description: returns profile data of user.

Syntax: getData(string User)

Return value: array.

```
$data = $uapi->getData ("sheiko");
```

updateData ()

Description: change profile data of user or group.

Developer Handbook

Environment Variables

Environment variables are available in templates and DDC though call:

```
<sapi:apply name="variable.value" />
```

or:

```
&variable.value;
```

Instructions `&variable.value;` are processed before others and can be used as parameters for XML Sapiens expressions and calls.

Following variables are available:

args_length – number of address line arguments (for example, in this case `/arg1/arg2/` variable will return 2);

extraargs_length – number of address line optional arguments (for example, in this case `/arg1/arg2/data/extrarg1/` variable will return 1);

DirectAccessArgs_length – number of arguments when record is requested (for example, in this case `/arg1/arg2/0000192/0000193/` variable will return 2);

document_url – the address of current document (for example, `gallery/`);

args.0 – the first argument of address line (for example, `gallery`). Any next argument is available in variables like `args.index_of_argument`;

extraargs.0 – the first argument of optional address line (for example, `something`). Any next argument is available in variables like `extraargs.index_of_argument`;

argsstring – screened string of address line arguments (for example, `gallery/`);

argsstringwithoutslash – the same, but without slashes (for example, `gallery`);

admin_view_http_path – address of administrative panel templates root folder (for example, `http://sapidcmf.rh8.rg/views/default/`);

delivery_view_http_path – address of site templates root folder (for example, `http://sapidcmf.rh8.rg/views/delivery/`);

http_path – current site address (for example, `http://sapidcmf.rh8.rg/`);

admin_http_path – administrative panel address (for example, http://sapidcmf.rh8.rg/admin/);

system_configuration_date – framework configuration date (for example, 2007/08/22);

system_version – framework version (for example, Build 85);

user_id – authorized user identifier (for example, pass);

site_id – open site identifier (for example, 1);

record_id – requested record identifier. If record is not requested it returns 0;

user_fullname – full name of authorized user;

user_login – login of authorized user;

user_profile – description template of authorized user (for example, default.tpl);

document_title – title of the open document (for example, Gallery);

document_template – template of the open document (for example, gallery.tpl);

document_id – identifier of the open document (for example, 9);

document_data_x – content data, wherein x is name of a content field;

record_data_x – content record, wherein x is name of a content field.

Content Queries (QC)

Content queries define how content within the document/record/user profile will be requested.

Sample of QC call:

```
1100101  
1100101  
1100101  
1100101  
1100101  
<sapi:apply name="qc.dataid.value" type="type" title="Caption" index=""  
src="" parent="" display="" />
```

wherein:

Index – points if query must be indexed;

Src – [rs:5] [root:catalogue/] [dbtable:xx] – points at source for SELECT query;

Parent – Parent ID for SELECT query;

Display – none if content must not be showed on site, if content is meant to be just kept to used somewhere as variable (for example, meta tags).

Following types of queries are already defined in the framework:

inputtext – line query (like <input type=text>);

article – query with WYSIWYG;

file – file query;

image – image query;

date – date query;


select – list query (like <SELECT> of HTML);

checkbox – yes/no.

radio – yes/no.

Expressions

XML Sapiens expressions are used in DDC conditions or in the templates though construction:



```
<sapi:apply exp="expression()" />
```

You can write in plugin and use you own expressions or use ready expression of the framework:

showselectvalue() – returns value of QC SELECT index;

showthumb(var, width, height, alt, extra, extraparams) – shows/creates thumbnail;

showthumb_background – the same with background;

showimage(var,alt,extra) – shows image in the list;

enlarge(var,v1) – enlarge value;

set(var,v1) – set value of an environment variable;

isequal(var,v1,v2) – if condition is correct returns v1, otherwise v2;

isenvnotnull(index,v1,v2) – if environment variable 'index' is not null returns v1, otherwise v2;

isnotnull(var,v1,v2) – if var is not null если var returns v1, otherwise v2;

datef(format, date) – transforms date;

enum(i, a, b, ...) – returns element with index i;

lt(a,b,...) – returns true if a < b;

leq(a,b,...) – returns true if a <= b;

gt(a,b,...) – returns true if a > b;

geq(a,b,...) – returns true if a >= b;

filter {*line*} – filters the list. Sample: AND struct.price='112';

filterfunction {*line*} – assigns user function to filter the list.

Sample:

DDC section:

```
<sapi:param name="filterfunction">checkfilter</sapi:param>
Секция plugins/index.php
function checkfilter($res) {
    foreach($res as $index => $fetch) {
        $res[$index]["entry_name"] = $fetch["entry_name"]." prefix";
    }
    return $res;
}
```

Enumeration variables:

this.length.value – number of enumeration elements;

this.this.name.value, this.this.entry_name.value – record title;

this.this.unified_id.value – record ID;

this.this.level.value – level (it is counted from parent document);

this.this.type.value – record type (record/recset);

this.this.cdate.value – creation date in DATETIME;

this.this.cdatetime.value – modification date in UNIX TIMESTAMP;

this.this.mdate.value – modification date in DATETIME;

this.this.href.value – record URL;

this.this.counter.value – record counter;

this.this.recsetlength.value – number of all nested records;

this.this.field.value – additional index field value;

this.this.parent_name.value – title of the parent element;

this.this.parent_href.value – URL of the parent element.

pagination()

Description: returns pagination data fo specified list

```
<sapi:for-each select="pagination()" name="enum">
<sapi:params>
<sapi:param name="pagination_id">&this.enum.pagination_id;</sapi:param>
<sapi:param name="type">pagination_frame</sapi:param>
</sapi:params>
<sapi:ifempty>Nothing found</sapi:ifempty>
<sapi:fallback>CMS-application error</sapi:fallback>
```

Parameters:

pagination_id – list ID;

type – pagination type: pagination or pagination_frame (frame length is defined in rc.conf.php).

Enumeration variables:

this.this.title.value – title;

this.this.range.value – title like 20-30;

this.this.href.value – link;

this.this.current.value – flag if it is current page.

&this.enum.length; – list length;

&this.enum.list_length; – selecting length;

&this.enum.pages_number; – page number;

&this.enum.range; –number of records on a page;

&this.enum.page_next_href; – URL of the next page;

&this.enum.page_prev_href; – URL of the previous page;

&this.enum.page_prev_number – number of the previous page;

&this.enum.page_next_number – number of the next page;

&this.enum.page_first_href; – URL of the first page;

&this.enum.page_first_number; – number of the first page;

&this.enum.page_last_href; – URL of the last page;

&this.enum.page_last_number; – number of the last page;

&this.enum.page_first_range; – title of the first page like 10-20;

get_track()

Description: returns track to current document

```
<sapi:for-each select="get_track()" name="enum" title="Get track">
```

Parameters:

none

Enumeration variables:

this.this.item_id.value – document ID;

this.this.item_level.value – nesting level;

this.this.item_type.value – document type (item/recset);

this.this.var.value – document address variable;

this.this.cdate.value – creation date in DATETIME;

this.this.cdatetime.value – creation date inUNIX TIMESTAMP;

this.this.mdate.value – modification date in DATETIME;

this.this.href.value – document address;

this.this.name.value – document title;

this.this.currentpage.value – active element.

get_search ()

Description: returns search results

```
<sapi:for-each select="get_search()" name="enum" title="Get channel">  
<sapi:params>  
<sapi:param name="keywords">&_get_keywords.value;</sapi:param>  
</sapi:params>  
<sapi:ifempty>Nothing found</sapi:ifempty>  
<sapi:fallback>CMS-application error</sapi:fallback>
```

Parameters:

keyword– search phrase.

Enumeration variables:

Appendix

Support

If you meet any troubles within framework installation or at the time of its exploitation, tell us about

Community site: Developer Handbook

Environment Variables

Environment variables are available in templates and DDC though call:

```
<sapi:apply name="variable.value" />
```

or:

```
&variable.value;
```

Instructions `&variable.value;` are processed before others and can be used as parameters for XML Sapiens expressions and calls.

Following variables are available:

args_length – number of address line arguments (for example, in this case `/arg1/arg2/` variable will return 2);

extraargs_length – number of address line optional arguments (for example, in this case `/arg1/arg2/data/extrarg1/` variable will return 1);

DirectAccessArgs_length – number of arguments when record is requested (for example, in this case `/arg1/arg2/0000192/0000193/` variable will return 2);

document_url – the address of current document (for example, `gallery/`);

args.0 – the first argument of address line (for example, `gallery`). Any next argument is available in variables like `args.index_of_argument`;

extraargs.0 – the first argument of optional address line (for example, `something`). Any next argument is available in variables like `extraargs.index_of_argument`;

argsstring – screened string of address line arguments (for example, `gallery/`);

argsstringwithoutslash – the same, but without slashes (for example, `gallery`);

admin_view_http_path – address of administrative panel templates root folder (for example, `http://sapidcmf.rh8.rg/views/default/`);

delivery_view_http_path – address of site templates root folder (for example, `http://sapidcmf.rh8.rg/views/delivery/`);

http_path – current site address (for example, `http://sapidcmf.rh8.rg/`);

article – query with WYSIWYG;

file – file query;

image – image query;

date – date query;


select – list query (like <SELECT> of HTML);

checkbox – yes/no.

radio – yes/no.

Expressions

XML Sapiens expressions are used in DDC conditions or in the templates though construction:



```
<sapi:apply exp="expression()" />
```

You can write in plugin and use you own expressions or use ready expression of the framework:

showselectvalue() – returns value of QC SELECT index;

showthumb(var, width, height, alt, extra, extraparams) – shows/creates thumbnail;

showthumb_background – the same with background;

showimage(var,alt,extra) – shows image in the list;

enlarge(var,v1) – enlarge value;

set(var,v1) – set value to an environment variable;

isequal(var,v1,v2) – if condition is correct returns v1, otherwise v2;

isenvnotnull(index,v1,v2) – if environment variable 'index' is not null returns v1, otherwise v2;

isnotnull(var,v1,v2) – if var is not null returns v1, otherwise v2;

datef(format, date) – transforms date;

enum(i, a, b, ..) – returns element with index i;

lt(a,b,...) – returns true if a < b;

leq(a,b,...) – returns true if a <= b;

gt(a,b,...) – returns true if a > b;

geq(a,b,...) – returns true if a >= b;

filter {*line*} – filters the list. Sample: AND struct.price='112';

filterfunction {*line*} – assigns user function to filter the list.

Sample:

DDC section:

```
<sapi:param name="filterfunction">checkfilter</sapi:param>
Секция plugins/index.php
function checkfilter($res) {
    foreach($res as $index => $fetch) {
        $res[$index]["entry_name"] = $fetch["entry_name"]." prefix";
    }
    return $res;
}
```

Enumeration variables:

this.length.value – number of enumeration elements;

this.this.name.value, this.this.entry_name.value – record title;

this.this.unified_id.value – record ID;

this.this.level.value – level (it is counted from parent document);

this.this.type.value – record type (record/recset);

this.this.cdate.value – creation date in DATETIME;

this.this.cdatetime.value – modification date in UNIX TIMESTAMP;

this.this.mdate.value – modification date in DATETIME;

this.this.href.value – record URL;

this.this.counter.value – record counter;

this.this.recsetlength.value – number of all nested records;

this.this.field.value – additional index field value;

this.this.parent_name.value – title of the parent element;

this.this.parent_href.value – URL of the parent element.

pagination()

Description: returns pagination data fo specified list

```
<sapi:for-each select="pagination()" name="enum">
<sapi:params>
<sapi:param name="pagination_id">&this.enum.pagination_id;</sapi:param>
<sapi:param name="type">pagination_frame</sapi:param>
</sapi:params>
<sapi:ifempty>Nothing found</sapi:ifempty>
<sapi:fallback>CMS-application error</sapi:fallback>
```

Parameters:

pagination_id – list ID;

type – pagination type: pagination or pagination_frame (frame length is defined in rc.conf.php).

Enumeration variables:

this.this.title.value – title;

this.this.range.value – title like 20-30;

this.this.href.value – link;

this.this.current.value – flag if it is current page.

&this.enum.length; – list length;

&this.enum.list_length; – selecting length;

&this.enum.pages_number; – page number;

&this.enum.range; –number of records on a page;

&this.enum.page_next_href; – URL of the next page;

&this.enum.page_prev_href; – URL of the previous page;

&this.enum.page_prev_number – number of the previous page;

&this.enum.page_next_number – number of the next page;

&this.enum.page_first_href; – URL of the first page;

&this.enum.page_first_number; – number of the first page;

&this.enum.page_last_href; – URL of the last page;

&this.enum.page_last_number; – number of the last page;

&this.enum.page_first_range; – title of the first page like 10-20;

get_track()

Description: returns track to current document

```
<sapi:for-each select="get track()" name="enum" title="Get track">
```

Parameters:

none

Enumeration variables:

`this.this.item_id.value` – document ID;

```
this.this.item_level.value – nesting level;
```

`this.this.item_type.value` – document type (item/reset);

`this.this.var.value` – document address variable;

this.this.cdate.value – creation date in DATETIME;

`this.this.cdatetime.value` – creation date in UNIX TIMESTAMP;

`this.this.mdate.value` – modification date in DATETIME;

`this.this.href.value` – document address;

this.this.name.value – document title:

`this.this.currentpage.value` – active element.

get_search ()

Description: returns search results

```
<sapi:for-each select="get_search()" name="enum" title="Get channel">
<sapi:params>
<sapi:param name="keywords">&_get_keywords.value;</sapi:param>
</sapi:params>
<sapi:ifempty>Nothing found</sapi:ifempty>
<sapi:fallback>CMS-application error</sapi:fallback>
```

Parameters:

keyword– search phrase.

Enumeration variables:

Appendix

Support

If you meet any troubles within framework installation or at the time of its exploitation, tell us about

Community site: <http://www.sapid-club.com>