|  |
| --- |
| **System Design Specifications** |
| API Development Portal  for  API Gateway Server  Document Version 1.10 16-th, April 2014 | | |



Copyright © Axway Software, 2014

All rights reserved.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the copyright owner, Axway Software.

This document, provided for informational purposes only, may be subject to significant modification. The descriptions and information in this document may not necessarily accurately represent or reflect the current or planned functionalities of this product. Axway Software may change this publication, the product described herein, or both. These changes will be incorporated in new versions of this document. Axway Software does not warrant that this document is error free.

Axway Software recognizes the rights of the holders of all trademarks used in its publications.

# Preface

Purpose of this DOCUMENT

The purpose of this document is to list the technical requirements and the implementation details of the API Portal project.

Related documentation

This Document is related and refers to the following sources:

**APIDeveloperPortal.docx**

Intended Audience

The intended audience of this document is everyone involved in the project implementation, deployment, acceptance or management. It is expected that the reader is familiar with all base products and systems related to it as well as the targeted business requirements.

# Table of Contents

[1 Preface 2](#_Toc382493379)

[2 Table of Contents 3](#_Toc382493380)

[3 Revision History 4](#_Toc382493381)

[4 Overview 6](#_Toc382493382)

[4.1 Prerequisites 6](#_Toc382493383)

[4.2 Deliverables 6](#_Toc382493384)

[4.3 Assumptions 6](#_Toc382493385)

[4.4 Requirements 7](#_Toc382493386)

[4.5 Architectural Overview 8](#_Toc382493387)

[5 Use Cases 10](#_Toc382493388)

[5.1 User Activities 10](#_Toc382493389)

[5.1.1 Self registration 10](#_Toc382493390)

[5.1.2 User Login 11](#_Toc382493391)

[5.1.3 Self management 12](#_Toc382493392)

[5.1.4 Testing APIs through the API Catalog or API Tester page 14](#_Toc382493393)

[5.1.5 Monitoring 15](#_Toc382493394)

[5.1.6 User Blog Use Case 16](#_Toc382493395)

[5.1.7 User Forum Use Case 16](#_Toc382493396)

[5.1.8 Application List Use Case 16](#_Toc382493397)

[5.1.9 New Application Use Case 17](#_Toc382493398)

[5.1.10 Edit Application Use Case 18](#_Toc382493399)

[5.1.11 Delete Application Use Case 20](#_Toc382493400)

[5.2 Admin Activities 21](#_Toc382493402)

[5.2.1 Add Documentation 21](#_Toc382493403)

[5.2.2 Moderate Blogs and Forums 21](#_Toc382493404)

[5.2.3 Configure API Dev Portal 22](#_Toc382493405)

[6 Other features 23](#_Toc382493406)

[6.1 Blog – K2 23](#_Toc382493407)

[6.2 Forum 23](#_Toc382493408)

[6.3 Deletion of users with E-mails that are not confirmed yet. 24](#_Toc382493409)

[7 Appendix A 25](#_Toc382493410)

[7.1 Setting and getting user attributes 25](#_Toc382493411)

[7.1.1 Get (obtain the list of custom attributes) 25](#_Toc382493412)

[7.1.2 Set (set the value of a custom attribute) 25](#_Toc382493413)

[7.2 Performing REST calls with PHP 26](#_Toc382493414)

# Revision History

| **Revision** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 1.0 | 24-th February 2014 | Yordan Nedelchev | First draft. |
| 1.1 | 6-th of March 2014 | Yordan Nedelchev | - Appendix with samples added  - Description of the login use case added  - Description of the OAuth use case added  - Additional assumptions added. |
| 1.2 | 7-th of March 2014 | Yordan Nedelchev | Additional Assumptions |
| 1.3 | 10-th of March 2014 | Totyo Totev | Additional Use Cases added. |
| 1.4 | 10-th of March 2014 | Yordan Nedelchev | Update some diagrams. |
| 1.6 | 12-th of March 2014 | Yordan Nedelchev | Updated the design according to the comments from David.  Additional information regarding the Blog and Forum functionality included. |
| 1.7 | 13-th of March 2014 | Yordan Nedelchev | Adding links to web resources regarding requirements and installation about  Joomla, PHP, etc… |
| 1.8 | 13-th of March 2014 | Totyo Totev | Adding screenshots regarding listing, creation and editing of applications. A minor change in Delete Application use case is added. |
| 1.9 | 7-th of April 2014 | Yordan Nedelchev | Add more details regarding the list of APIs how it will be displayed and how do we put static content that gives more details for a given API. |
| 1.10 | 16-th of April 2014 | Yordan Nedelchev | * Change the API Gateway Server version to 7.3 (beta), although just a beta version is available at the time of implementation. * Add note that security is not a concern for the connection between Joomla Portal and API Gateway. * Testing of the API performed through swagger will be proxied by the joomla portal and will surport API Key authentication and HTTP Basic Authentication only. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

in this chapter

This chapter briefly describes the features of the solution package.

## Prerequisites

Host Environment with

* Installed Apache Web Server **(**[**http://www.apache.org/**](http://www.apache.org/)**)**
* PHP Support **(**[**http://www.php.net/**](http://www.php.net/)**)**
* Installed MySQL database Server **(**[**http://www.mysql.com/**](http://www.mysql.com/)**)**
* Installed **Joomla 3.x (**[**http://www.joomla.org/technical-requirements.html**](http://www.joomla.org/technical-requirements.html)**)**
* API Gateway Server 7.3 (beta) installed and configured on the same or another machine.

Note: For development purposes it is quite easy to just install [XAMPP](http://www.apachefriends.org/index.html), which will automatically provide you with Apache + MySQL. The [Joomla](http://joomlacode.org/gf/download/frsrelease/19239/158104/Joomla_3.2.3-Stable-Full_Package.zip) could be directly installed on top of it or could be installed via [bitnami](http://bitnami.com/learn_more).

The editor’s choice IDE is [Netbeans](http://dlc.sun.com.edgesuite.net/netbeans/7.4/final/bundles/netbeans-7.4-php-windows.exe), but more or less any IDE or even a simple text editor can be used. And [this](http://wiki.netbeans.org/HowToConfigureXDebug#Overview) is how you configure netbeans for debugging with Xdebug.

## Deliverables

This is a list of the deliverables that will be provided by Axway

* Joomla plugin(s) such as joomla components and joomla modules that can be uploaded and installed in Joomla through the Joomla admin UI.
* Documentation and step by step buide regarding how to install and configure the plugin(s).

## Assumptions

* The API Dev Portal will be localized in English only.
* No explicit UI templates will be defined. One of the existing Joomla UI templates will be used. Axway provided package will use some of the basic Joomla templates, but then customer may implement or purchase from third parties another template on their owns. We also assume that any template purchased from third parties would follow the basic Joomla template standards.
* In general, the API Dev Portal will not offer any Change, Update or Delete functionality for objects Users or APIs. New APIs cannot be created.

Managing organizations is also not necessary, but management (Create, Update, Delete) of applications is necessary. Also it should be possible for a user to share applications with other members of their organization.

The only use case when we will support Creation of a usr is the creation of a new user by the Self-Registration page.

The only use cases when we have update of a user is the update of the user property that the user has accepted terms and conditions and the other one is when a user changes their own properties like E-mail or password.

Similar limitations apply for Joomla administrator as well. Joomla administrator will not be able to Create, Change, Delete APIs, Applications, Users, Organizations and so on…

* When the Joomla administrator uploads new content, we will support just plain text, html formatted content and a file. In the case of file, the dev portal will just store it and make it available for download – no need to parse content, reformat or similar. For example, administrator uploads some documentation in the form of a PDF and then it appears available for users to download it.
* Joomla Portal and API Gatway are supposed to be in a demilitarized or secure network zone, so we do not expect men-in-the-middle or other security attacks against the communication between them. Current solution will not guarantee prevention of such attacks. More specifically Joomla portal will trust any CA certificate when connecting to API Gateway, regarding of whether it is issued by trusted authority or not or even self signed.

## Requirements

Typically Axway customers will install and configure API Gateway Server to provide a set of network accessible interface APIs. These APIs are exposed for usage by their partners. Partners of our customers would typically automate the usage of the APIs in a program. So they need developers with development skills to program that automation. To do that, these developers need documentation about interface API, as well as examples about how to use it. This documentation and knowledge is best distributed in a web portal.

Several of our competitors have similar API Dev Portals, so clients are used to use them. In several deals customers have asked about such portal and the lack of it has even been a deal breaker sometimes.

Professional Services – Custom Development Group are required to build such portal so that we can ease our customer when they have to offer API management for their partners.

We should structure our portal similar to a content management system. The idea is that our customers define the APIs in API Gateway, and then they go and upload the documentation for it in the API Dev Portal.

The API Dev Portal should offer the following features:

|  |  |
| --- | --- |
| End User functionality   * + Self Registration   + API Listing   + Testing page   + Forums   + Blogs (Optional) | Admin Functionality   * API Dev Portal Configuration * Upload change content * Moderator of the Forums * Moderator of the Blogs |

Here are similar portals of our customers which we can use for reference.

* <https://developer.kp.org/>
* <https://developer.kp.org/>
* <http://developer.espn.com/>
* <http://developer.tomtom.com/>
* <https://developer.carepass.com/>
* <http://developer.cokecce.com/>

## Architectural Overview

The API Dev Portal is not installed on top of API Gateway Server. It is separately installed. It is entirely dedicated for the developers of the partners of our customers. So once they clear up how the API works and can be used, then they can directly access API Gateway and do not need to use the portal for a proxy.



# Use Cases

in this chapter

This chapter covers the main use cases this solution would cover.

## User Activities

This section describes the use cases related to user activities. Users are the partner developers who use the portal to register and read documentation from it.

### Self registration

**Prerequisites:**

API Dev Portal installed and configured

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** opens the API Dev Portal in their browser and goes to the self registration page. This page is only visible if the user has not signed it. The page should contain a [captcha](https://en.wikipedia.org/wiki/Captcha) mechanism to protect from automated account registration. This page also contains a list of the Terms and Conditions that the user must accept in order to use the API Dev Portal. If user agrees with the terms and conditions, they must check a checkbox in the page.
2. **Partner Developer** enters the required information like user name, password, E-mail, etc… and submits the form.
3. **API Dev Portal** accepts the submitted form and does initial validation. For example it validates the captcha entry to make sure there is a real person sitting accessing the portal and not just an automated process. It also validates that the user has accepted the terms and conditions.

If initial validation fails, then an error message is returned to the Partner Developer.

1. **API Dev Portal** uses REST API of API Gateway Server to send a user creation request. Along with the user creation request, there is also a request to set the custom user property named «TermsAndConditionsAccepted » to have value of «true».

For more information about how to set user attributes see [here](#_Setting_and_getting).

1. **API Gateway** receives the user creation request and tries to create a user in its database.
2. **API Gateway** returns the success/error status of that operation back to the API Dev Portal. If the creation is successful, then **API Dev Portal** would also send a confirmation E-mail to the E-mail specified by the user. The E-mail should have a link that points to some «Email confirmation service», so that when clicked the E-mail confirmation service will confirm the E-mail and finish user creation. Only at that point the user will be considered fully configured and allowed to login inside Joomla UI.
3. **API Dev Portal** returns the success/error status to the Partner Developer. So the partner developer will either see a page with an error or a page that explains them that initial creation is successful, but as a second step they need to check their E-mail and click on the link in order to confirm the E-mail validity.
4. **Partner Developer** checks their E-mail and clicks on the link that sends a requiest to the so called «E-mail confirmation service».
5. **Email confirmation service** confirms the E-mail and finishes user registration process. Then another E-mail is sent to the user informing them that the registration process has finished and they can proceed with login. A link to the login page will be provided.

### User Login

**Prerequisites:**

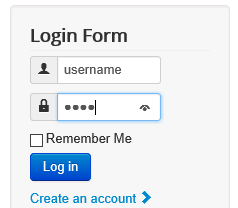
Partner Developer has already registered in the API Dev Portal

**Participants:**

Partner Developer, API Dev Portal, API Gateway

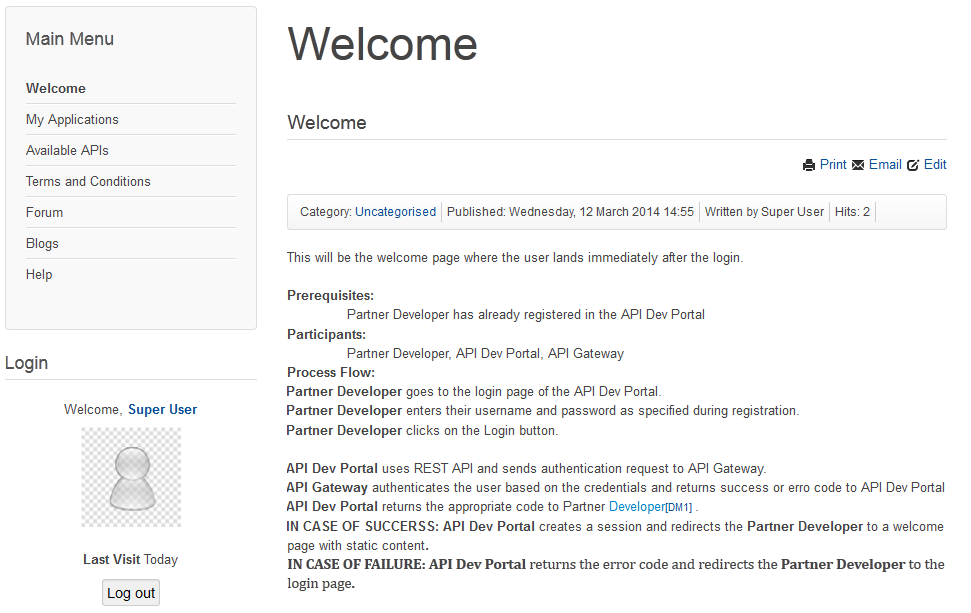
**Process Flow:**

1. **Partner Developer** goes to the login page of the API Dev Portal.
2. **Partner Developer** enters their username and password as specified during registration.
3. **Partner Developer** clicks on the Login button.



1. **API Dev Portal** uses REST API and sends authentication request to API Gateway.
2. **API Gateway** authenticates the user based on the credentials and returns success or erro code to API Dev Portal
3. **API Dev Portal** returns the appropriate code to Partner Developer.
4. **IN CASE OF SUCCERSS: API Dev Portal** creates a session and redirects the **Partner Developer** to a welcome page with static content**.**
5. **IN CASE OF FAILURE: API Dev Portal** returns the error code and redirects the **Partner Developer** to the login page**.**

Sample Welcome page:



### Self management

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to the page with their account properties, selects a property they want to enter. For example – the password.
2. **Partner Developer** selects a new value for the property.
3. **Partner Developer** submits the form.
4. **API Dev Portal** uses API Gateway REST API to initiate property change for the given user.
5. **API Gateway** receives the user update request and executes it.
6. **API Gateway** returns the success/error status of that operation back to the API Dev Portal.
7. **API Dev Portal** returns the success/error status to the Partner Developer.



### Testing APIs through the API Catalog or API Tester page

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

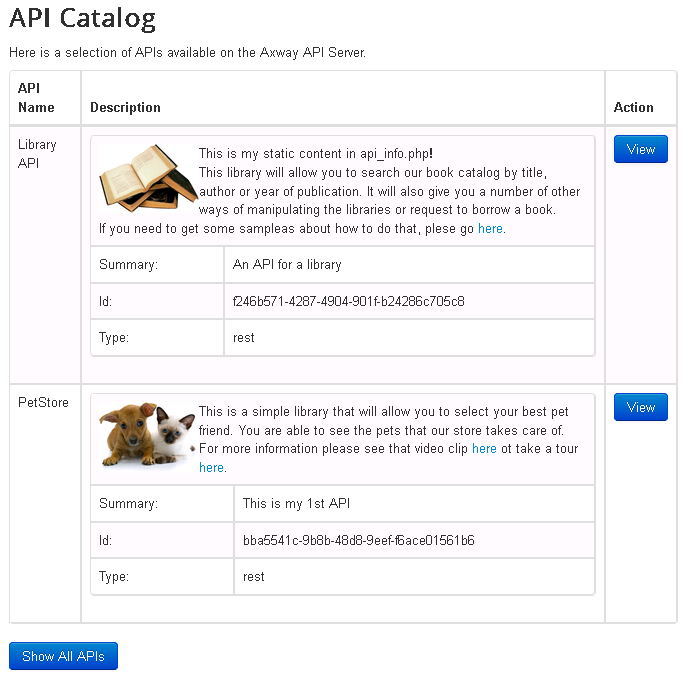
Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** opens the API Catalog page or API Tester page. Both pages list APIs that are available for usage by the Partner Developer. API Tester shows a complete list ast obtained from the API Gateway. API Catalog shows a pre-selected list of APIs only.

**The API Catalog should look similar to this:**

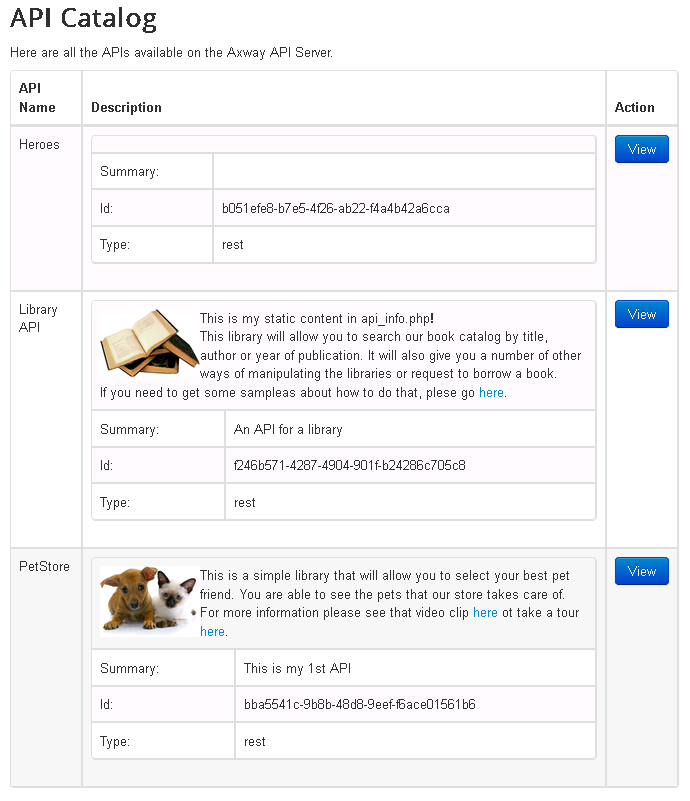
**See also [here](http://developer.espn.com/docs).**

****

Dynamicly retrieved API info.

Static content with images and links, etc…

There will be a configurable place for specifying a filter of fine grained APIs that will be shown at first. Then there will be a button that will allow the user to show all APIs. If the filter is not specified or is empty, then all APIs will be shown by default.



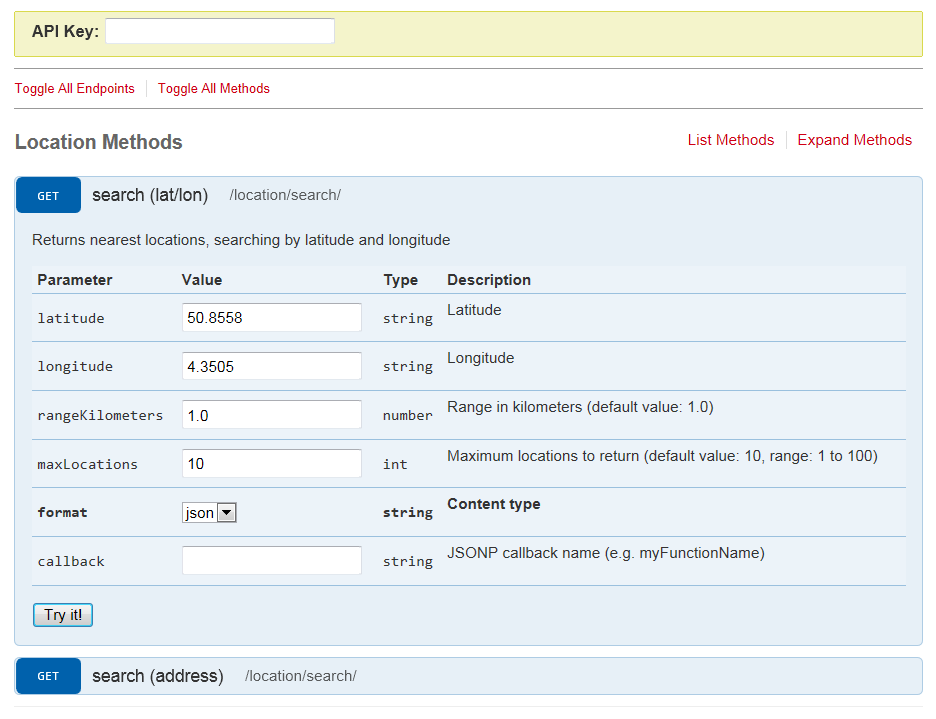
Inside the API list there would be placeholders for static fontent like images which will help users reasily remember or associate that API as well as static text with links to tutorials or additional static documenatation.

The static information will be defined per API. For each file there would be a corresponding set of static files to be included in different placeholders of the UI view. These files will be put in a folder which name would match to the id of the API. If files in this folder exist, they will be included in their appropriate places.

Along with that the dynamic properties of the APi

**Here is how API Testing Page will look like:**

**We plan to use the Swagger UI for it. See** [**here**](http://developer.cokecce.com/io-docs) **and** [**here**](http://swagger.wordnik.com/)**. Source of** [**swagger**](https://github.com/wordnik/swagger-core)**.**

**[](http://developer.cokecce.com/io-docs)**

Since swagger is Javascript based and is really executed on the browser, there can be cross browser issues if it tries to access the API Gateway directly. These issues can be resolved on the API Gateway server side it the server supports Access-Control-Allow-\* HTPTS headers. Unfortunately at the time of writing this documents the latest available version of API Gateway (7.3 beta) does not support this.

So the solution will be designed to proxy the connection between the client browser and the API Gateway (that is similar to a reverse proxy). In that way swagger functionality running on the client browser will always connect to Joomla portal site which will proxy the connection to the API Gateway and then will filter out the result to substitute any links to point back to the Joomla portal.

1. **Partner Developer** selects the api they want to test and enters appropriate input parameters for that API call. Swagger will also offer the user to fill in authentication credentials. These credentials correspond to he so called security deviced in API Gateway terminology.The exact list of security devices that will be supported by our solution are:
   1. **API Key Authentication**
   2. **HTTP Basic Authentication**
2. **Partner Developer** submits a test request via this API.
3. **API Dev Portal** accepts the submitted request and forwards it to the API Gateway using the same credentials.
4. **API Gateway** receives the API call and executes it.
5. **API Gateway** returns the output of the API call to the API Dev Portal.
6. **API Dev Portal** returns the output of the API call to the Partner Developer.

### Monitoring

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to the monitoring page.
2. **API Dev Portal** shows the monitoring page. That page display graphs of Application Usage and API Usage. Same capabilities as existing API Portal Monitoring page (No requirement to maintain look & feel of graphs).

### User Blog Use Case

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to their blog page and submits a new blog post.

### User Forum Use Case

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to the forum page, selects topic or creates a new topic. Then user reads the posts in the topic and optionally submits a new post for the given topic.

### Application List Use Case

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

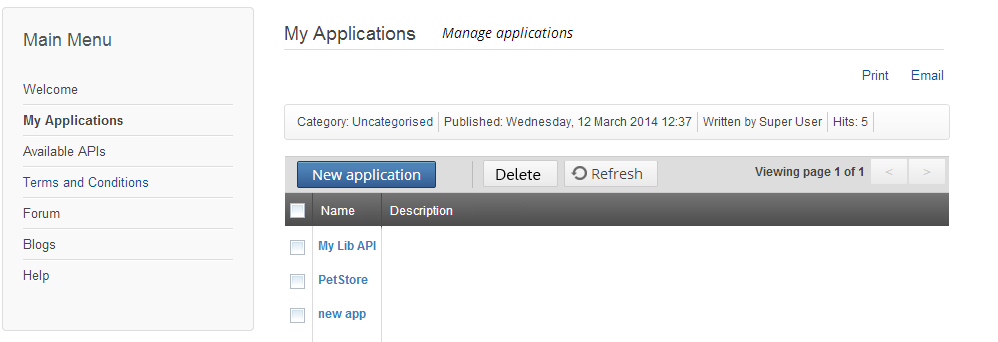
Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to the application listing page.
2. **API Dev Portal** uses the REST API of the API Gateway to send an application listing request.
3. **API Gateway** executes request and returns a list of all of the applications together with their properties.

**API Dev Portal** displays the information about the available applications.

Application listing page should look similar to this:



### New Application Use Case

**Prerequisites:**

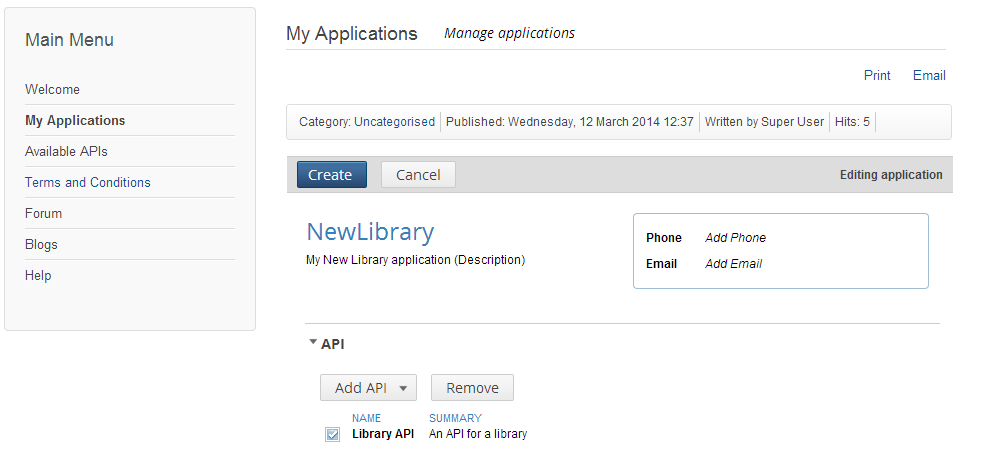
API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to the application listing page.
2. **Partner Developer** clicks “New Application” link/button
3. **Partner Developer** enters Application data in fields
4. **Partner Developer** clicks “Save Application” buttons/link
5. **API Dev Portal** uses the REST API of the API Gateway to send a *Create Application* request.
6. **API Gateway** executes request and returns
7. **API Dev Portal** uses the REST API of the API Gateway to send an application listing request.
8. **API Gateway** executes request and returns a list of all of the available applications together with their properties
9. **API Dev Portal** displays the information about the available applications.



### Edit Application Use Case

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

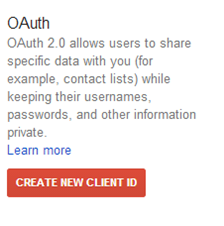
**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

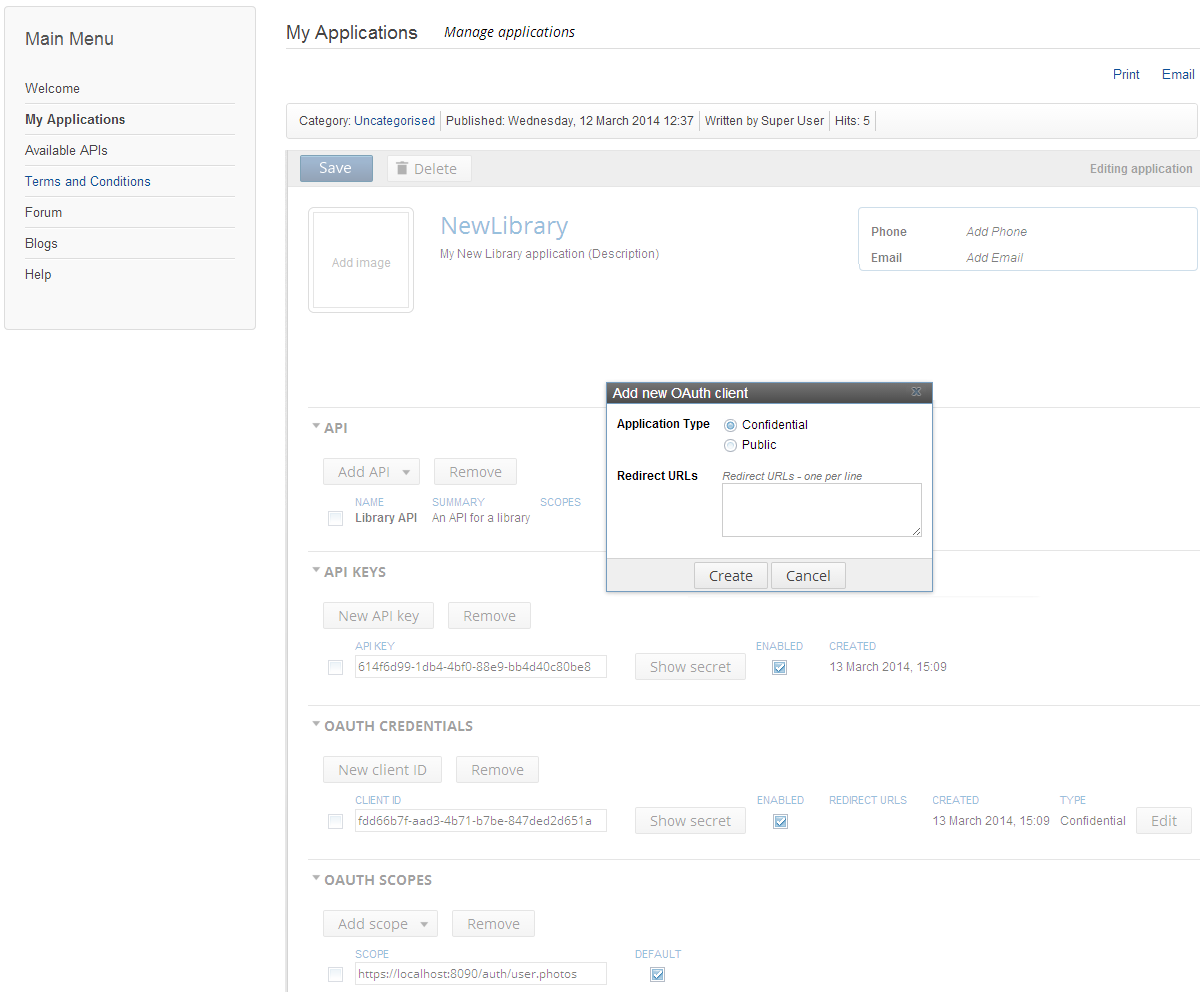
1. **Partner Developer** goes to the application listing page.
2. **Partner Developer** selects an application by clicking on it in application list. Inside Application properties, all “security credentials” settings should be visible in the application screen. As part of application properties there are also certain tasks that can be performed with an applicataion. One of them is the OAuth key generation. It will offer a generate button as well as explanation about usage of OAuth and a text field for the output.

**Here is an example of how the page should look like:**



**Partner Developer** may clicks on the key generation button, then **API Dev Portal** will use the REST API of the API Gateway to generate a new OAuth key for the given partner developer user. **API Gateway** willgenerate a new OAuth key and returns it to the **Partner Developer** inside the web UI.

1. **Partner Developer** changes Application data in fields.
2. **Partner Developer** clicks “Save Application” buttons/link
3. **API Dev Portal** uses the REST API of the API Gateway to send an *Update Application* request.
4. **API Gateway** executes request and returns
5. **API Dev Portal** uses the REST API of the API Gateway to send an application listing request.
6. **API Gateway** executes request and returns a list of all of the available applications together with their properties
7. **API Dev Portal** displays the information about the available applications.



### Delete Application Use Case

**Prerequisites:**

API Dev Portal installed and configured, Partner Developer has logged in the portal.

**Participants:**

Partner Developer, API Dev Portal, API Gateway

**Process Flow:**

1. **Partner Developer** goes to the application listing page.
2. **Partner Developer** selects the application(s) that need to be deleted through selecting the associated checkboxes or open preferred application for editing
3. **Partner Developer** clicks “Delete Application” button/link which will be available in application list page and application editing page
4. **Partner Developer** Confirms delete operations
5. **API Dev Portal** uses the REST API of the API Gateway to send a *Deletion Application* request.
6. **API Gateway** executes request and returns
7. **API Dev Portal** uses the REST API of the API Gateway to send an application listing request.
8. **API Gateway** executes request and returns a list of all of the available applications together with their properties.
9. **API Dev Portal** displays the information about the available applications

## Admin Activities

This section describes use cases related to administrator activities. Administrators are users that are locally defined in the API Dev Portal – they do not have a corresponding account on the API Gateway. They can moderate the forums and blogs and can add new content (documentation) in the API Dev Portal.

### Add Documentation

**Prerequisites:**

API Dev Portal installed and configured.

**Participants:**

Partner Developer, API Dev Portal, Administrator

**Process Flow:**

1. **Administrator** logs in locally in the API Dev Portal.
2. **Administrator** selects the API they want to add documentation to. Then the administrator uploads the documentation. The documentation is in the form of a file or an html markup. Documentation is per API call.
3. **API Dev Portal** accepts the documentation and stores it locally. It keeps an association between the given documentation and the corresponding API.
4. **Partner Developer** logs into the API Dev Portal. Now they can access the newly uploaded documentation.

### Moderate Blogs and Forums

**Prerequisites:**

API Dev Portal installed and configured.

**Participants:**

Partner Developer, API Dev Portal, Administrator

**Process Flow:**

1. **Administrator** logs in locally in the API Dev Portal.
2. **Administrator** selects a blog or topic in a forum and deletes or adds content into it. Administrator has an exclusive right to edit posts done by any Partner Developer user.
3. **API Dev Portal** accepts the change and stores it locally.

### Configure API Dev Portal

**Prerequisites:**

API Dev Portal installed.

**Participants:**

API Dev Portal, Administrator

**Process Flow:**

1. **Administrator** logs in locally in the API Dev Portal.
2. **Administrator** select section “Configuration -> API Dev Portal”.
3. **Specify host/ip, administrator name and password of the API GW**.

**Test connection and save the settings**

# Other features

in this chapter

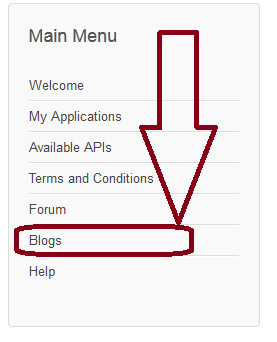
This chapter includes other features not described in the use cases – such like blogs and forums, etc…

## Blog – K2

The blog K2 that is available as Joomla plugin will be used.

See <http://extensions.joomla.org/extensions/authoring-a-content/content-construction/8061?qh=YTozOntpOjA7czoyOiJrMiI7aToxO3M6NDoiazIncyI7aToyO3M6MzoiJ2syIjt9>

The blog functionality would be available from the main menu:



The blog functionality should be integrated with the Joomla login mechanism for users.

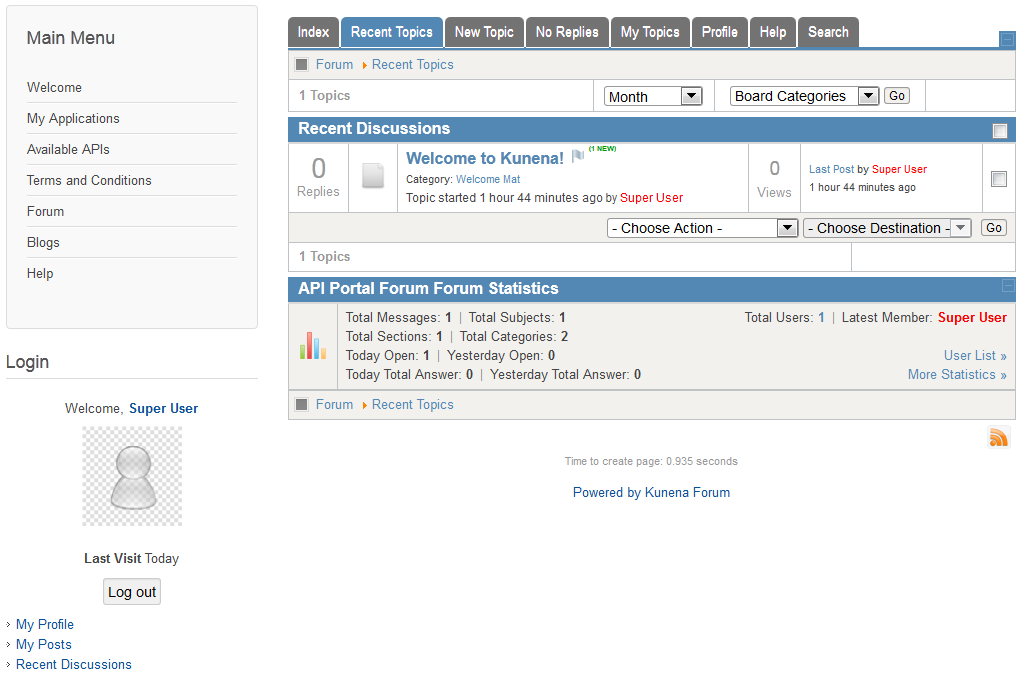
## Forum

The forum Kunena which is available as Joomla plugin will be used:

See <http://extensions.joomla.org/extensions/extension-specific/kunena-forum-extensions/13955?qh=YToyOntpOjA7czo2OiJrdW5lbmEiO2k6MTtzOjg6Imt1bmVuYSdzIjt9>

The forum functionality would be available from the main menu:

Here is a sample of how it would look like:



The forum should be integrated with the login mechanism supported by Joomla.

## Deletion of users with E-mails that are not confirmed yet.

Since we create new user accounts for users at the time when they fill in and submit the self-registration form but we do not allow them to login until they confirm their E-mail,

there is a possibility that an user enters a wrong E-mail address by mistake and is therefore never able to confirm it.

In order to not overload the system with such fake accounts we need to have some cleanup mechanism.

Not yet clear how we do that. Could be a cron job that starts at certain time intervals.

# Appendix A

in this chapter

This appendix provides more technical details on some of the items:

## Setting and getting user attributes

API Gateway provides a REST API for getting and setting user attributes. Custom attributes can be defined as well.

Till now the only custom attribute will be the status of acceptance of Terms and Conditions. It would be called «TermsAndConditionsAccepted» and should have value of either true or false.

Here are examples of REST calls to get and set user attributes:

### Get (obtain the list of custom attributes)

|  |
| --- |
| **POST /api/portal/v1.0/organizations**/ HTTP/1.1  **contenttype:** **application/json**  **Authorization:** **Basic YXBpYWRtaW5AbG9jYWxob3N0OmNoYW5nZW1l**  **UserAgent:Jakarta CommonsHttpClient/3.1**  **Host:** **localhost:8075**  **ContentLength:** **145**  {  **"name"** : "MyOrg",  **"description"** : "My organization.",  **"phone"** : "+353 (1) 6742000",  **"email"** : "myorg@axway.com",  **"TermsAndConditionsAccepted"** : "false",  **"enabled"** : true  } |

### Set (set the value of a custom attribute)

|  |
| --- |
| **PUT /api/portal/v1.0/organizations/c85cf2e6cb5e4f37afb25f0d250e40f2** HTTP/1.1  **contenttype:** **application/json**  **Authorization:** **Basic YXBpYWRtaW5AbG9jYWxob3N0OmNoYW5nZW1l**  **UserAgent:** **Jakarta CommonsHttpClient/3.1**  **Host:** **localhost:8075**  **ContentLength:** **238**  {  **"id"** : "c85cf2e6cb5e4f37afb25f0d250e40f2",  **"name"** : "MyOrg",  **"description"** : "My organization.",  **"phone"** : "+353 (1) 6742000",  **"email"** : "org2@axway.com",  **"TermsAndConditionsAccepted"** : "true",  **"enabled"** : true,  **"restricted"** : false,  **"createdOn"** : 1367424410635  } |

## Performing REST calls with PHP

PHP does not have a built-in support for calling REST APIs, however several libraries exist that can do that. Our suggestion is to use the [pest](https://github.com/educoder/pest) library.

See [this](http://www.code-thrill.com/2013/11/php-rest-client-with-pest.html) link for some more usage info.

Here is a short usage example:

First copy the library under

*[JOOMLAHOME]*/components/*com\_apiportal*/lib/pest/

And then

|  |
| --- |
| **include\_once** **'lib/pest/PestJSON.php'**;  **try** {  $**address** = "**http://10.128.16.70/**";  $**pest** = **new** PestJSON($**address**);  $**result** = $**pest**->get("/api/ping?"  ."sessionKey="  ."api29aca822eaf99f60e2aeb6ad6fa3736c2644065230cc1daa07"  ."&format=json");  **echo** **'<pre>'** . **print\_r**($**result**, **1**) . **'</pre>'**;  } **catch** (Exception $**e**) {  **echo** $**e**->getMessage();  } |