

Moovur – Development Documentation

Using the Moovur API for incorporating Mollom for Joomla

First published: February 2009, version 1.0
Moovum - www.moovum.com

Moovur Development Documentation

A guide to implement Mollom into Joomla Extensions using Moovur

Copyright

The information, text, names, images, photos, logos and Icons are distributed as is under the license terms listed below. The publication goes without any warranty for commercial and practical use without violating any rights. We are not responsible for legal implications of our content.

We grant you permission to re-use this information under the terms of the [Common Creative Attribution-Noncommercial-Share Alike 3.0 license](#), see also details below. If you create a translation of this document please let us know. The authors of the content represent their own mind. In now way the content is linked to other sites, organizations or individual unless explicitly stated.

Moovum

Moovum is the company that provides web services and web applications based upon the Joomla Application framework and the Joomla content management system. One of the services that is provided by Moovum is Moovur. Moovum is legal copyright holder of all material provided, that is the code, documentation and support provided via www.moovum.com under terms of use presented on the web site, or material provided.

Moovur

Moovur is a Joomla 1.5 native extension containing a Joomla 1.5 native component and several Joomla 1.5 native plug-ins that implements the Mollom anti-spam service into the core distribution of Joomla. Moovur is not only an extension that implements Mollom, it also is an Application Programming Interface (API) for third party developers that makes it very easy to implement Mollom in Joomla 1.5 extensions.

More information about Moovur can be found at : <http://www.moovum.com/products/moovur>

Mollom

The web is changing. User contribution is now what makes or breaks a site. Allowing users to react, participate and contribute while still keeping your site under control can be a huge challenge. Mollom is a web service that helps you identify content quality and, more importantly, helps you stop spam on your blog, social network or community website. When site moderation becomes easier, you have more time and energy to interact with your web community.

More information about Mollom can be found at : <http://www.mollom.com>

Table of Contents

1.Preface.....	4
2.Moovur architecture.....	7
2.1.Implementation methods.....	8
3.Implementation of Moovur in Joomla Extensions.....	11
3.1.Direct implementation in the extension code.....	11
3.2.Implementation using the Moovur plug-in architecture.....	12

1. Preface

This document will guide you through the basic concepts of Moovur. We will explain the architectural principles of the Joomla extension and also explain how you can develop your own solution to implement Mollom into your Joomla site.

Moovur has been build especially for making implementation of Mollom into Joomla extensions. It is possible to implement Mollom without the use of Moovur, but this will take considerable more time then re-using the Moovur API. Mollom is a generic service, Moovur has been created as a flexible Joomla API around the Mollom service, when done right you only need around 10 lines of code to activate Mollom into your extension!

We won't cover the concepts of Mollom. There are some excellent pieces of documentation available on the Mollom site, if you want to know more about Mollom please take a closer look at the following documentation:

- Brief description on how Mollom works : <http://mollom.com/how-mollom-works>
- Mollom technical white paper : <http://mollom.com/files/mollom-technical-whitepaper.pdf>
- Mollom API documentation : <http://mollom.com/files/mollom-client-api.pdf>

What does this document cover?

Chapter 2, *Moovur architecture*, gives a description about the Moovur architecture. We describe the component and plug-in architecture, and how you can implement Mollom using the Moovur API. There are two methods you can use, and both are described in detail here.

Chapter 3, *implementation of Moovur in Joomla extensions*, gives a technical description on how the two methods can be implemented. We give a description, including coding examples and best practices.

Appendix 1, *Moovur error codes*. In this appendix we give an explanation on the error codes that can be generated by Moovur, or the error code standards that need to be implemented when you implement Moovur into your extension.

What do you need for this document?

The target audience of this document are *Joomla developers*. To use this documentation effective you need to have proper knowledge of the Joomla application and content management (CMS) framework. With that you need to be a PHP developer, preferable someone who has build one or more extensions for Joomla 1.5.

All coding examples are based upon PHP 5.2 and MySQL 4.1 or newer versions of both PHP and MySQL. Obviously you also need to have a working version of Apache or Lighttpd, to make it easy...a working Joomla 1.5 installation running PHP 5.2 or MySQL are required.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this document, what you liked or may have disliked. Reader feedback is important for us to develop better documentation that you really get the most out of.

To send us general feedback, simply drop a message in our support forum making sure to mention a reference to this documentation and a clear subject in the forum title. Remember that you need to register to the Moovum site before you can send in feedback.

If there is any additional information that you would like us to publish, please send us a note in the support forum with a clear description of your request. We will try to answer your questions and will consider to fulfill your request.

When there is any topic that you have expertise in and you are interested in either writing or contributing back, feel free to also contact us using the support forum.

Download of Moovur

Moovur can be downloaded from <http://www.moovum.com/products/moovur/download>. Moovur is written under GPL version 2 and for that you have free access to the source code and code examples mentioned in this documentation.

If you implement Mollom using Moovur for your extension, please share back the results to us so we can incorporate it in the official version of Moovur. As we like to share Moovur for free with you, we would appreciate that you do the same and share back your results with us.

Errate

Although we have taken every care to ensure the accuracy of the contents of this documentation, mistakes do happen. If you find a mistake in this documentation – this might be a mistake in the text or code examples – we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help improve subsequent versions of this documentation. If you find any errata, please report them in our support forum. Be as specific as you can be. Once your errata has been verified, your submission will be accepted and the errata will be added to the list of existing errata. Ultimately all the feedback will be processed into a new version of the development documentation.

Questions

You can post your question into our support forum if you are having a problem with some aspect of this documentation, and we will do our best to address it.

Conventions used in this document

In this documentation, you will find numbers of styles that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

There are two styles for code. Code words in text are shown as following: “When we define `$var` the `get($var, $default);` method, the get method makes sure that the variable is instantiated properly”.

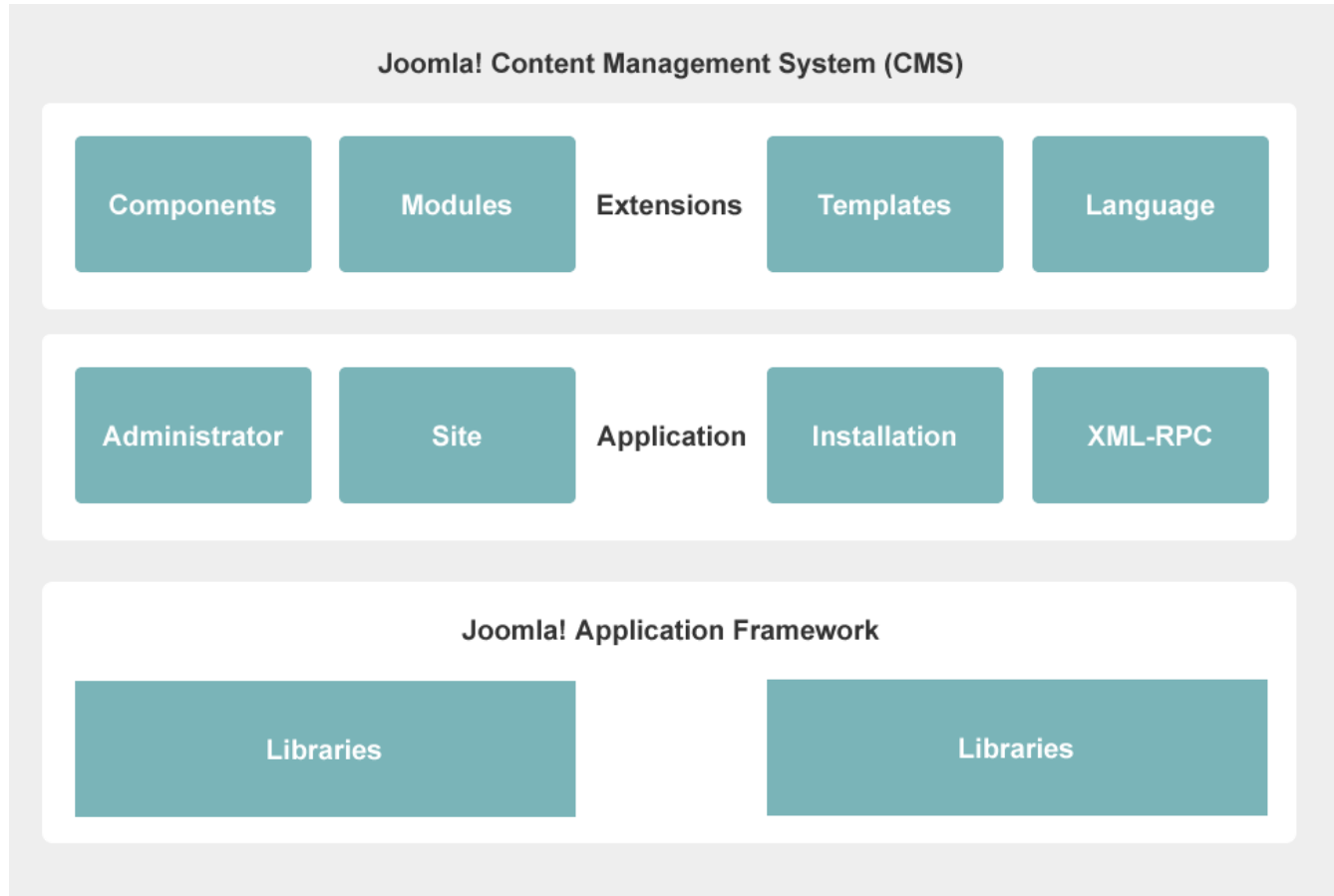
A block of code will be set as follows:

```
/*
 * Get method
 *
 * @param string name of the variable we want to get the value from
 * @param mixed default value for variable when not found
 * @return mixed get value
 */
public function get($var, $default = null) {
    return $this->registry->get($var, $default);
}
```

For the actual coding conventions we refer to the “Joomla coding style and standards” that can be found on the Joomla documentation wiki. The Joomla coding style and standards can be found at http://docs.joomla.org/Coding_style_and_standards

2. Moovur architecture

To get a proper understanding on how the Moovur architecture looks like, we would like to start with an architectural overview of the Joomla CMS. Below you see an architectural overview of the Joomla CMS.



When you install Moovur several pieces of software are installed, a short summary:

- **Moovur component (com_moovur).** This is the main component where Moovur is configured and the general Moovum management like log review, latest version check etc.
- **Core Moovur plug-in (moovur.php).** This plug-in implements the Moovur API. A new plug-in category “Moovur” is created, this plug-in also manages the behavior of all additional Moovur plug-ins using the Joomla dispatcher.
- **Moovur Contact plug-in (moovur_com_contact.php).** This is a plug-in that extends the default behavior of the contact forms in the core distribution of Joomla by implementing the Mollom service.
- **Moovur User plug-in (moovur_com_user.php).** This plug-in extends the default behavior of the user management logic like login and, registration.

The behavior of the plug-ins can be controlled by the plug-in properties that can be managed via the Joomla back-end. Some of the general behavior is defined by the settings in the Moovur component.

2.1. *Implementation methods*

Mollom is the generic anti-spam service that can be accessed using the generic Mollom class that is provided by Mollom. As extension developer you can manually implement this generic Mollom class for every form as Mollom can be integrated in PHP applications. Moovum has created a flexible and powerful Joomla API that makes it incredible easy to implement Mollom into Joomla extensions. To put it simple: Moovur implements a Joomla native API that makes is incredible easy to implement Mollom.

When we designed Moovur we used the main design principles of Joomla; reduce complexity. We fully make use of the object oriented and pattern based approach that has been implemented into Joomla. We also have tried to give the developers as much as possible control for implementing Mollom into their extensions. For this we have created two methods:

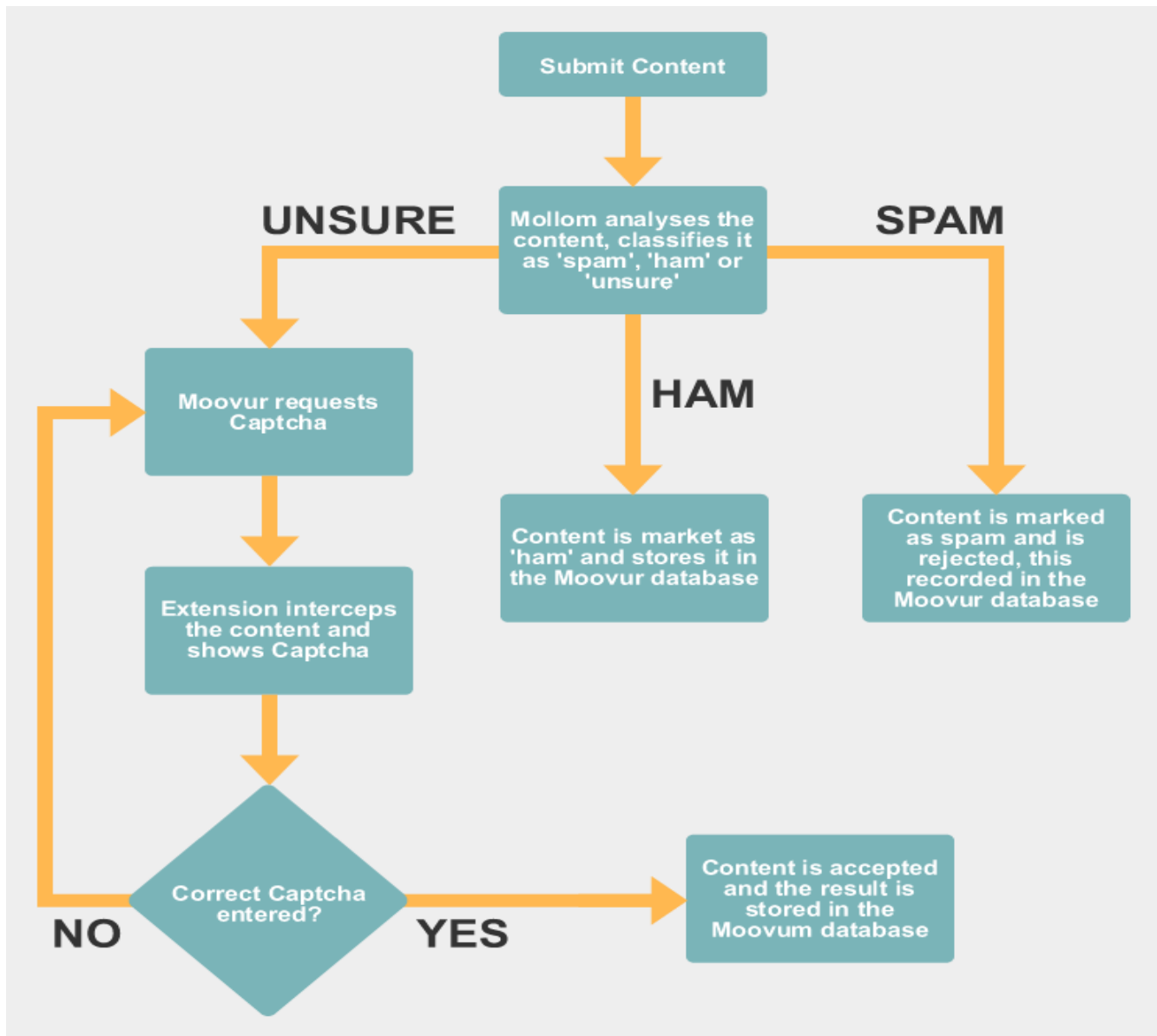
1. **Direct implementation in the extension code.** The advantage of this method is that with very few lines of code you will be able to implement Mollom using the Moovur API. This logic will be part of the extension logic.
2. **Implementation using the Moovur plug-in architecture.** Moovur provides a possibility to incorporate Mollom anti-spam protection if for some reason the extension developer sees no need to implement Mollom into the extension. All it takes is to write a Joomla plug-in within the “Moovur” plug-in class family.

We prefer extension developers implement Moovur using the first option. It's the most descent way to implement because it creates no specific overhead. It is up to the extension developer to determine which options of the Mollom solution can be influenced (the so called properties). When the first method is used, by default the settings from Moovur will be used, any specific behavior needs to be implemented by the extension developer.

In this chapter we will cover both methods in detail. We have tried to create an API that bridges between the Mollom service and the Joomla application framework to make it as easy as possible to implement this service in your Joomla application. With the documentation in this chapter, and the examples provided in the Moovur extension you will be able to implement Mollom in a short period of time (between 1-2 hours). Feel free to contact the Moovur development team if you are an extension developer and have questions regarding Moovur or the documentation provided.

Mollom behavior

In the Mollom technical white-paper (<http://mollom.com/files/mollom-technical-whitepaper.pdf>) and the Mollom client API 1.0 documentation (<http://mollom.com/files/mollom-client-api.pdf>) the intended behavior of the Mollom service is described. We give a brief overview of this behavior because it is essential to understand what it takes to implement this behavior into your extension using one of the available Moovur methods.



Moovur makes it possible to implement the desired Mollom behavior, but the actual implementation depends on the Moovur implementation method you select.

Direct implementation in the extension code

When you implement Mollom directly into your extension using Moovur, make sure you implement the intended behavior of visitors first posting content, then send the result to Mollom for analyses and act based upon the response.

This implementation method gives you the freedom to implement the Captcha handling in any other way, not that it would make sense in our opinion, but you can do it. Please check out the technical details of the implementation in the next chapter.

Implementation using the Moovur plug-in architecture

Moovur implements a Joomla specific API by installing the Moovur plug-in as system plug-in. This plug-in is triggered after the framework has loaded and initialized and the router has route the client request. Routing is the process of examining the request environment to determine which component should receive the request. This component optional parameters are then set in the request object to be processed when the application is being dispatched.

To put it simple; all it does is instantiate the “moovur” plug-in class, of course only for those extension that are activated. Every Moovur plug-in needs to implement two methods for activating Mollom for this extension. With this implementation you don't have to alter the core code of the extension where you want to implement Mollom into.

Within the Moovur distribution we provide two plug-in extensions that implement Mollom into the Joomla core distribution.

3. Implementation of Moovur in Joomla Extensions

3.1. *Direct implementation in the extension code*

This is the preferred way to implement Mollom into your extension since it is the most easy and clean way. We tried to create an API that is as simple as possible, and we are convinced we achieved exactly that.

Below you will find the example code to validate the content. The object values of course need to be filled with the actual values of the content you want to validate.

```
if(class_exists('Moovur')) {
    $obj = new stdClass();
    $obj->title = 'The title data, if available';
    $obj->text = 'The full text, if available';
    $obj->author_name = 'The authors name, if available';
    $obj->author_url = 'URL, if available';
    $obj->author_email = 'email, if available';
    $obj->author_id = 'Joomlas user id, if available';
    $obj->author_openid = 'Openid, if available (not used)';
    Moovur::checkContent($obj, 'component', 'task');
}
```

This is the way you submit the content as visualized in the previous chapter. From here Moovur will take care of the validation and form handling. When the content is not marked as spam, the interception logic of Moovur will simply trigger the actual task for the extension.

If you want to add Captcha, you simply can use the following code.

```
if(class_exists('moovur')) {
    echo Moovur::getCaptcha('component.task');
}
```

Please note that if you are checking the content from the same form, don't add Captcha, it will be added by Moovur itself when needed.

3.2. *Implementation using the Moovur plug-in architecture*

As you can see implementing Mollom into your extension can be very easy. We have implemented a second method if you want to implement it without touching the core code of the extension. All it takes is a Joomla plug-in that belongs to the 'moovur' class.

Below we show part of the code of the contact plug-in as it is provided within the Moovur core package. We provide two plug-ins and those contain a more detailed example than provided in this documentation, especially parameter behavior (plug-in properties)

```
<?php
defined('_JEXEC') or die('Restricted access');
class plgMoovurMoovur_Com_Contact extends Jplugin
{
    /*
     * Constructor
     */
    public function __construct(&$subject, $options = array()) {
        parent::__construct($subject, $options);
    }

    /*
     * Check response for user plug-in. Within this method we include all
     * possible actions and handle them if active.
     */
    public function moovurCheckResponse() {
        ...
        ...
        ...
    }

    /*
     * Perform the actual form validation. Based upon the action requested we
     * will insert the Captcha.
     */
    public function moovurValidateForm(&$form) {
        ...
        ...
        ...
    }
}
```

When a form is submitted, the Moovur API will intercept and fire the `moovurCheckResponse()` Method. The form fields that need to be validated need to be handled in the plug-in, and the API is called in the way we described it in the previous paragraph.

The `moovurValidateForm()` method is used to insert Captcha into an existing form by performing a search and replace on the submit button. The actual statement to get a successful result depends on the form you try to implement this plug-in for.

Appendix 1: Error codes Moovur

- 66501 - Error getting statistics
- 66502 - Routine server update failed
- 66503 - Error initializing Mollom class
- 66504 - Error checking content via Mollom
- 66505 - Error sending feedback to Mollom
- 66506 - PHP5 required
- 66507 - Could not publish 3pd plug-ins on install/upgrade
- 66508 - Could not publish core plug-in on install/upgrade
- 66509 - Could not find XML setup file
- 66510 - Could not unpublish 3pd plug-ins on install/upgrade
- 66511 - Could not unpublish core plug-in on install/upgrade
- 66512 - Log table does not exist
- 66513 - Could not verify keys with mollom
- 66514 - error saving configuration
- 66515 - Error fetching captcha
- 66516 - Invalid input to Moovur::checkContent
- 66517 - Unexpected return from Mollom in checkContent
- 66518 - Error in checkContent
- 65519 - Curl or fsockopen required

Appendix 2: Error codes Moovum

66401 - Unknown error reported by Mollom
66402 - Invalid headers returned from Mollom
66403 - No body returned by Mollom call
66404 - unknown Curl error returned
66405 - No more servers available
66406 - Unknown FSocketOpen error returned
66407 - FSocketOpen infinite loop detected
66408 - Illegal method call to Mollom::doCall()
66409 - Public Key not configured
66410 - Private Key not configured
66411 - Server List not populated
66412 - Invalid response for checkContent
66413 - Invalid arguments for checkContent
66414 - Invalid response for checkCaptcha
66415 - Invalid response for getAudioCaptcha
66416 - Invalid response for getImageCaptcha
66417 - Invalid response for getServerList
66418 - Invalid response for getStatistics
66419 - Invalid argument for getStatistics
66420 - Invalid response for sendFeedback
66421 - Invalid argument for sendFeedback
66422 - Invalid response for verifyKey
66423 - Attempt to set invalid variable name
66424 - Expecting Numeric
66425 - Expecting String
66456 - Expecting Array