

OohLaLog User Guide

Copyright Notices

Copyright © 2015 OohLaLog Data, LLC. All rights reserved.

OohLaLog® is a registered trademark of OohLaLog Data, LLC in the United States and other countries.

OOHLALOG DATA, LLC
800 CONCAR DRIVE SUITE 100
SAN MATEO CA 94402
TEL: 650.358.5000
FAX: 650.358.5001
WWW.OOHLALOG.COM

GETTING STARTED.....	5
CREATE AN ACCOUNT.....	5
ADD AN APPLICATION.....	6
OPTION 1—COPY THE OLL API KEY INTO YOUR APPLICATION.....	7
OPTION 2—IMPORT SAMPLE LOG DATA.....	9
CREATE COUNTERS, RECIPIENTS AND ALARM NOTIFICATIONS.....	10
Setup a Counter.....	10
Setup an Alarm.....	12
Add Recipients.....	13
PLUG-IN DOCUMENTATION.....	14
RUBY.....	14
Insert API Key.....	14
Run the <i>bundle</i> Command.....	15
Install OohLaLog Gem.....	16
Verify Application.....	16
See Also.....	16
Next Step.....	16
JAVA.....	17
Installation.....	17
Configuration.....	17
Logging Context.....	18
Counters in Log4j.....	18
GRAILS.....	19
Add Oolalog to BuildConfig.groovy.....	19
Modify Config.groovy.....	19
Verify Application.....	20
Grails Artefact Logging and Counting.....	21
See Also.....	21
Next Step.....	21
.NET.....	22
Usage.....	22
Asynchronous Logging.....	22
Synchronous Logging.....	23
See Also.....	24
Next Step.....	24
PHP.....	25
Add Values to Global Variables.....	25
Add OohLaLog API Key.....	25
Limitations.....	26
See Also.....	26
Next Step.....	26
PYTHON.....	27
Pypi Plug-in Install.....	27
Add API Key.....	27
(Optional) Verify Log Forwarding.....	28
See Also.....	29
Next Step.....	29
JAVASCRIPT.....	30
Usage.....	30
Low Level APIs.....	31
See Also.....	31
Next Step.....	31
ORACLEDB.....	32

SYSLOG 33
 Configuration..... 33
 See Also 34
 Next Step..... 34



Getting Started

Get started with OohLaLog using these steps:

1. [Create an Oohlolog account.](#)
2. [Add an application.](#)
3. [\(Option 1\)](#) Import log data from your application into OohLaLog by copying the OLL API key into your application using our plug-in documentation.
4. [\(Option 2\)](#) Import sample log data.
5. Once log data appears in your dashboard, [create counters, alarms and alarm notifications.](#)

Create an Account.

The first step is to create your OohLaLog account. Go to the OohLaLog.com website and click **START 30 DAY FREE TRIAL**. Fill out the form and click **CREATE FREE ACCOUNT**.

GET STARTED!
Sign up for 30 days free trial. No credit card required.

First Name: _____
Last Name: _____
Email: _____
Password: _____
Confirm Password: _____
API Key: _____
Application Name: _____
Application URL: _____
Application Description: _____
Application Version: _____
Application Category: _____
Application Status: _____

CREATE FREE ACCOUNT

By clicking, you agree to our Terms & Conditions and Privacy Policy.

Create Free Account Form

Add an Application

Once an account is created the **New Application** page displays. Enter the Name of your application. There are no rules for this name.

Other details about your application are optional.

Options for LANGUAGE are one of the nine OohLaLog-supported languages: Ruby, Java, Grails, .NET, PHP, Python, Javascript, or OracleDB. Syslog is also supported.

Click **Save**

The screenshot shows a web form titled "New Application". At the top, there is a "Name" input field. Below this is a section titled "Details" which contains three optional fields:

- Repository URL (optional)**: The input field contains "https://github.com/username/app". A tooltip below it reads: "(Optional) Enter your projects repository web address. We can use this to help you better dive into your log messages."
- Issue Management URL (optional)**: The input field contains "https://github.com/username/app/issues". A tooltip below it reads: "(Optional) Add your issue management uri for quickly being able to open issues from your log messages."
- Language**: The input field contains "i.e. Ruby, Groovy, Java, Python". A tooltip below it reads: "(Optional) Set your applications language to gain syntax highlighting right in your log exceptions where relevant."

At the bottom right of the form are two buttons: "Save" (highlighted in blue) and "Close".


Add New Application

To see log data in your Dashboard and begin setting up counters, alarms and recipients for alarms, either (Option 1) send your application logs to OohLaLog using our plug-ins or (Option 2) import sample data.

Option 1–Copy the OLL API Key Into Your Application

OohLaLog performs “live” logging which allows logging events to be viewed as they occur from a running application or system. Application or system log data is graphed and visualized so you can identify trends that can lead to quickly solving application efficiency problems or system failures.

Send your application logs to OohLaLog using the API key that displays. Copy it and insert into your application using a plug-in.



You're almost done!










Here is your API key. It's also in the app settings. See the documentation section for more info.

API Key `0d14d322-cbce-40c0-aa80-bdd2b539d79e`

The API Key

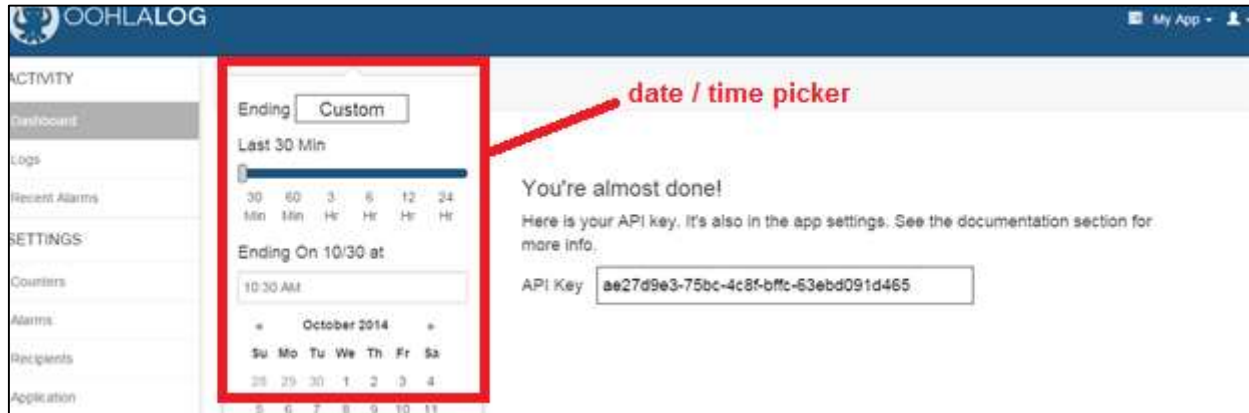
On the side bar menu, click **Documentation** and select one of nine plugins depending on the language for your application. See the [plug-in instructions](#) that tell how to insert your OLL API key and plug-in code into your application.

Available Plugins

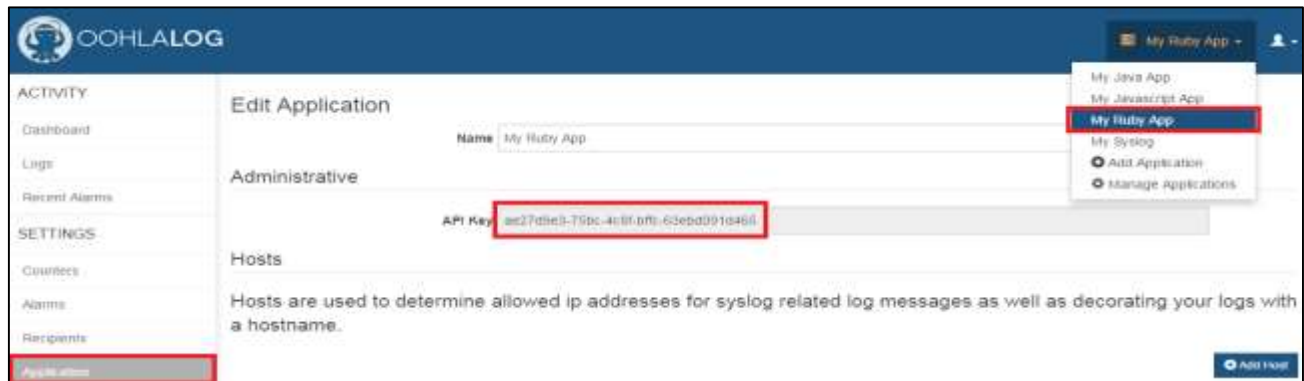
 Ruby Released 4/2/2014 Version 1.1.2 Documentation View Source	 Java Released 4/2/2014 Version 0.1.3 Documentation View Source	 Grails Released 1/1/2014 Version 0.2.3 Documentation View Source	 .NET Released 1/1/2014 Version 1.0.0 Documentation View Source
 PHP Released 3/1/2014 Version 1.0.0 Documentation View Source	 Python Released 4/2/2014 Version 0.0.2 Documentation View Source	 Javascript Released 1/1/2014 Version 1.0.0 Documentation View Source	 OracleDB Released 4/2/2014 Version 1.0.0 Documentation View Source
 Syslog Released 4/2/2014 Version 0.0.1 Documentation View Source			

Select a Plug-in for Your Application

Within a few minutes you should be able to go to **Dashboard** (by clicking in the left menu bar) and see a graphical representation of sample data for your application. You can now set the time/date for events captured.



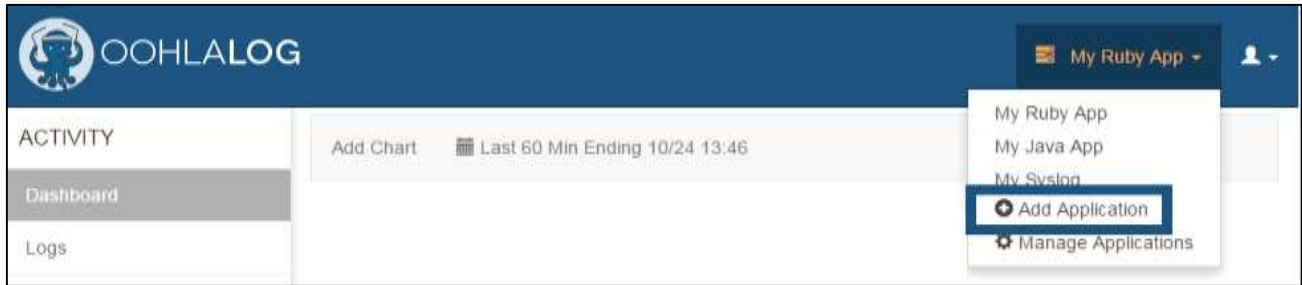
TIP: If you need to find this API key again use the pull-down menu in the upper right-hand corner to select your application name.



Finding Your API Key



TIP: If you did not create an application when you created an account, or want to create another application, use the pull down menu on the upper right corner and select **Add Application**.



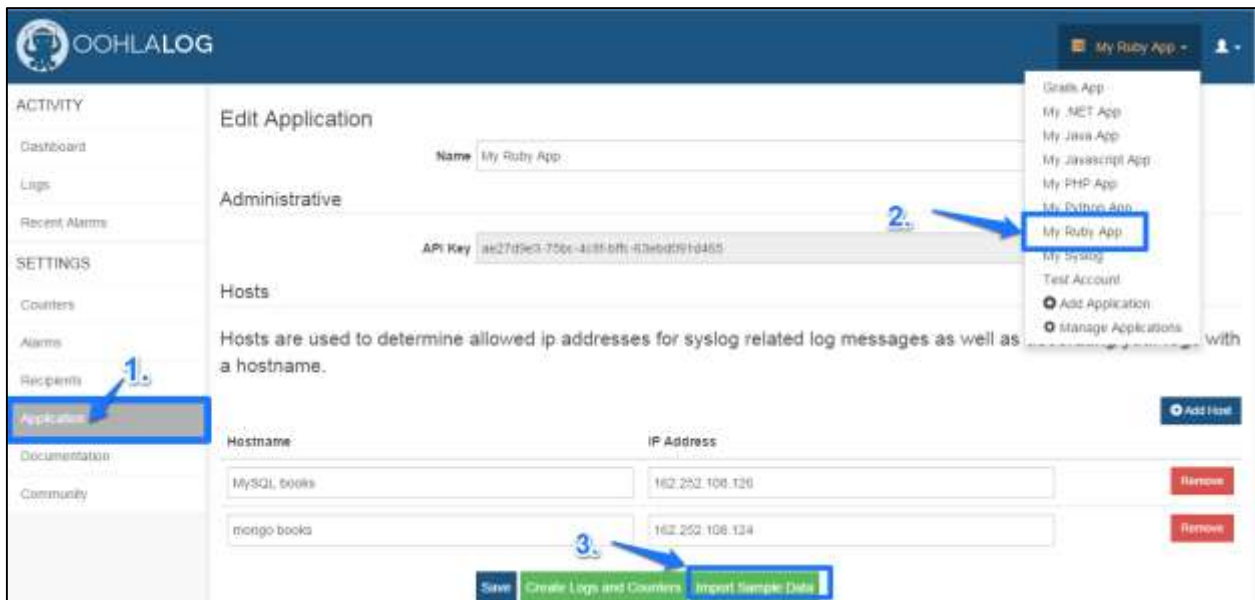
Add or Manage Applications

Select **Manage Applications** if you wish to edit or delete an OLL application.

Option 2—Import Sample Log Data

Use these steps to view sample data in OohLaLog’s dashboard without adding an API Key into your application.

1. Click **Applications** from the left menu
2. Select the application from the pull down menu.
3. On your application page, click **Import Sample Data** then click **OK** to confirm.



Steps to Import Sample Data

Within a few minutes you should be able to go to **Dashboard** by clicking in the left menu bar and see a graphical representation of sample data for your application.

Create Counters, Recipients and Alarm Notifications

Counters, alarms, recipients work in concert. First a counter is established. (A counter counts how many times a word or phrase appears in the application's log during a given time.) When an alarm threshold for that counter is reached, specified recipients receive email and/or SMS messages.

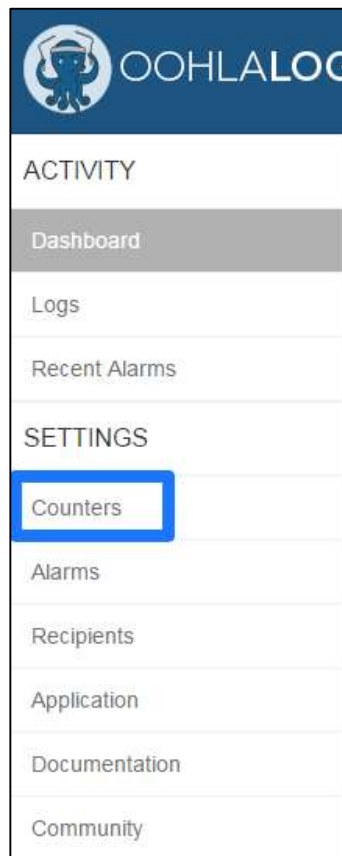
Custom counters are used to monitor business metrics in real time, keep track of what's important, and alert the right people to fix problems as they arise.

Here are just a few other things custom counters can do:

- Track how many times a user logs in
- Tell how many customers sign up
- Track specific errors and alert specified recipients

Setup a Counter

To create a counter, select Counters from the left side-menu under SETTINGS.




Choose Counters From Menu

Next, select your application from the pull-down menu in the upper right-hand corner. In this example “My Ruby App” is selected.



Select Your Application

Click the button  found on the upper right-hand corner. Fill in the information for a New Custom Counter.

A screenshot of a web form titled "New Custom Counter". The form contains several input fields and a priority slider. The "Name" field contains "Too many 'Howdys'", and the "Category" field contains "Debugging". Below these is a "Priority" slider with labels "ERROR", "WARN", "INFO", "DEBUG", and "TRACE", where "DEBUG" is selected. Under the "Filter Options" section, the "Log Pattern" field contains "Howdy", with a note below it: "If a log matches this regular expression, the counter is incremented." To the right of this note is a red-bordered button labeled "Advanced". Below the "Log Pattern" field are three empty text boxes for "Level Regex", "Category Regex", and "Stack Trace Regex". At the bottom, there is a "Match Mode" dropdown menu with "Any" selected and a blue highlight.

Fill in New Counter Information

Name can be any name you want to give your counter. Category is a name for a counting group that makes sense to you. The Priority slider selects the priority level for this counter. In our example, the label, Debug, will appear next to every “Howdy” in the log.

Log Pattern is filled in using regular expression (regex) format. Our example counts the number of times “Howdy” appears in our Ruby App log. Click Advanced to use more regex filtering options.

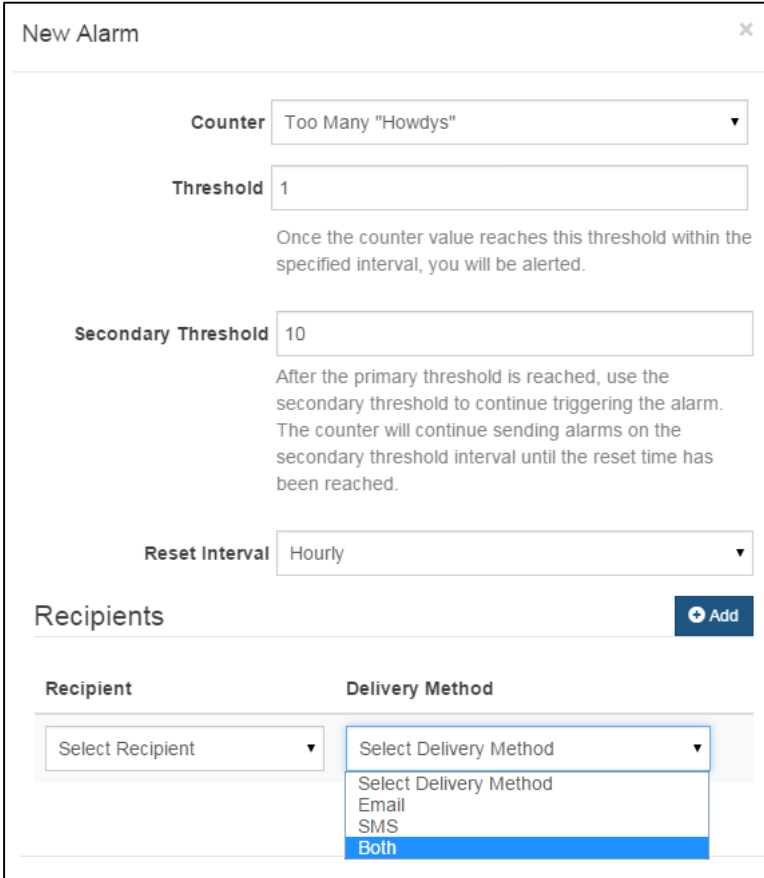
Here’s a quick start guide to regex.

<http://www.regular-expressions.info/quickstart.html>

Setup an Alarm

An alarm is used to trigger a messages when counter thresholds are reached. To create an alarm select Alarms from the left-hand menu.

Click  found on the upper right-hand corner. When the New Alarm form appears fill in the information.



New Alarm [Close]

Counter Too Many "Howdys" [Dropdown]

Threshold 1 [Input]
Once the counter value reaches this threshold within the specified interval, you will be alerted.

Secondary Threshold 10 [Input]
After the primary threshold is reached, use the secondary threshold to continue triggering the alarm. The counter will continue sending alarms on the secondary threshold interval until the reset time has been reached.

Reset Interval Hourly [Dropdown]

Recipients [Add]

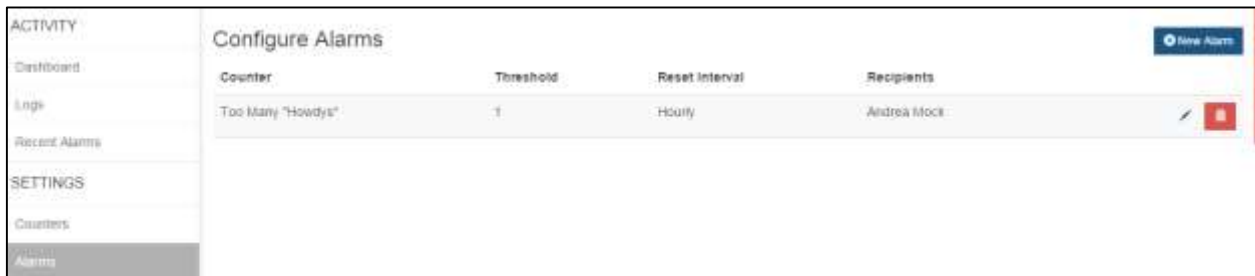
Recipient	Delivery Method
Select Recipient [Dropdown]	Select Delivery Method [Dropdown] Select Delivery Method Email SMS Both

Add New Recipients

The Reset Interval can be Hourly, Daily, Weekly, Monthly, Yearly or Never.



TIP: Select Alarms from the left menu to see a list of your alarms.



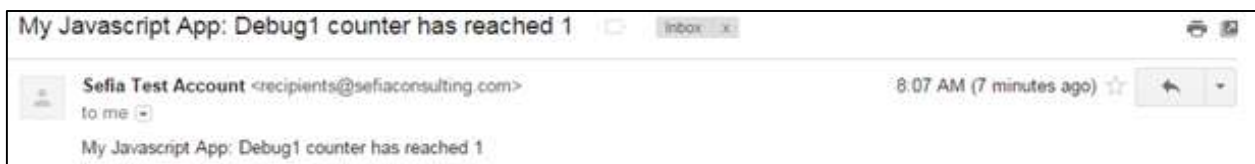
View Alarm List

Add Recipients

To add a new recipient, click  then fill in the New Recipient form.

New Recipient Form

This example shows the type of email message a recipient receives.



Email Message Example

Plug-in Documentation

This section includes the plug-in documentation for these languages and system logs:

- [Ruby](#)
- [Java](#)
- [Grails](#)
- [.NET](#)
- [PHP](#)
- [Python](#)
- [Javascript](#)
- [OracleDB](#)
- [Syslog](#)

Ruby

Before you begin you have set up your Ruby application in OohLaLog and copied its API key. Next you will follow these steps to begin sending your application log files to OohLaLog:

1. Insert the API key into your Ruby application's *application.rb* file.
2. Run the Ruby *bundle* command.
3. Install the OohLaLog gem into your application.
4. Verify that your Ruby application is running.

Insert API Key

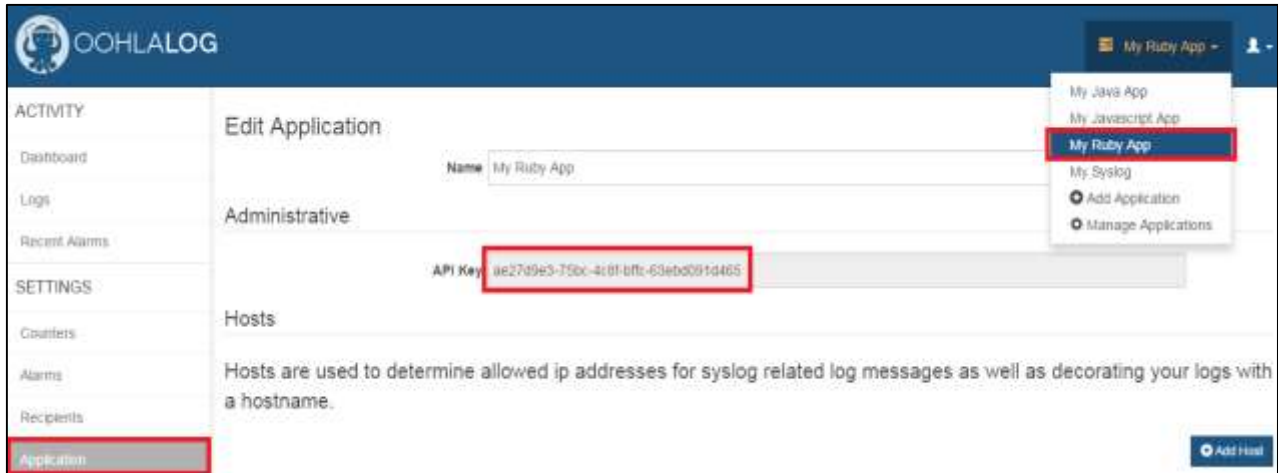
In your application's *application.rb* file, enter the OohLaLog API key as highlighted.

```
//Example add OohLaLog API key to application.rb file.
```

```
require File.expand_path('../boot', __FILE__)
require 'rails/all'
Bundler.require(:default, Rails.env)
module <Your App Name Here>
  class Application < Rails::Application
    Oohlalog.api_key = "8bd553fd-f059-4c07-9115-d9e1bc595a1f"
    config.assets.initialize_on_precompile = false
  end
end
```



TIP: To find your Ruby application's API key, select **Application** in the left-menu. Use the pull-down menu in the upper right-hand corner to select your application, for instance "My Ruby App." See this application's API key in the **Administrative** area.



Your Application's API Key

Run the *bundle* Command.

Run the Ruby *bundle* command:

//Example running bundle command.

```
Source 'https://rubygems.org'  
gem 'rails', '4.0.0'  
gem 'oohlalog'  
gem 'sass-rails', '~> 4.0.0'
```

Here's a sample of the expected output after the *bundle* command is executed.

//Example expected output for my_rails_app.

```
ubuntu@ubuntu-VirtualBox:~/my_rails_app$ bundle  
Using rails 4.0.0  
Using oohlalog 1.2.2  
Using sass-rails 4.0.3  
Your bundle is complete!
```



TIP: Use *bundle show [gemname]* to see where a bundled gem is installed.

Install OohLaLog Gem

This example shows an OohLaLog gem installed into the application, `my_rails_app`.

```
//Example OohLaLog gem installed in "my_rails_app".
```

```
ubuntu@ubuntu-VirtualBox:~/my_rails_app$ gem install oohlalog
Fetching: oohlalog-1.2.3.gem (100%)
Successfully installed oohlalog-1.2.3
1 gem installed
Installing ri documentation for oohlalog-1.2.3...
Installing RDoc documentation for oohlalog-1.2.3...
```

Verify Application

The following shows that the `my_rails_app` is running.

```
//Example running my_rails_app.
```

```
ubuntu@ubuntu-VirtualBox:~/my_rails_app$ rails s
=> Booting WEBrick
=> Rails 4.0.0 application starting in development on http://0.0.0.0:3000

=> Run `rails server -h` for more startup options
=> Ctrl-C to shutdown server
[2014-10-24 21:20:27] INFO WEBrick 1.3.1
[2014-10-24 21:20:27] INFO ruby 1.9.3 (2013-02-22) [i686-linux]
[2014-10-24 21:20:27] INFO WEBrick::HTTPServer#start: pid=4034 port=3000
```

See Also

For more information, see <https://rubygems.org>.

Next Step

After sending Ruby logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).

Java

OohLaLog provides a log4j appender that can be utilized by any Java application to push logs into the cloud. Several features are provided right out of the box.

Installation

The log4j appender is stored in a maven/nexus repository that can be easily included into your application's dependencies.

First, to your dependency resolver list, add the Bertram Nexus Repo found at:

<http://nexus.bertramlabs.com/content/repositories/publicReleases/>

Next add this maven dependency to your project

```
<dependency>
  <groupId>oohlalog</groupId>
  <artifactId>oohlalog-4j</artifactId>
  <version>0.2.3</version>
</dependency>
```

NOTE: It may be necessary to add GSON as an application dependency. By default the maven jar file will attempt to resolve GSON, but if it cannot, you may want to specify this dependency yourself.

Configuration

Configuration for OohLaLog is straightforward. There are a few ways to configure log4j depending on your Java application. You can follow the properties based configuration or the XML based configuration.

```
log4j.appender.oohlalog=com.oohlalog.log4j.OohLaLogAppender
log4j.appender.oohlalog.layout=org.apache.log4j.PatternLayout
log4j.appender.oohlalog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-
5p %c{1}:%L - %m%n
```

#Required: Replace with OohLaLog instance Api Key

```
log4j.appender.oohlalog.AuthToken=cf27826d-b08a-45ef-bdc8-3b11384a3638
```

#Optional: number logs to buffer before posting to OLL (lower numbers impact app performance)

```
#log4j.appender.oohlalog.MaxBuffer=100
```

```
#Optional: age of logs in buffer before automatic posting to OLL (lower
numbers impact app performance)
#log4j.appender.oohlalog.TimeBuffer=10000
```

NOTE: Not all of these configuration options are mandatory. The primary ones to be concerned about are the appender class path and the authToken properties.

Logging Context

The OohLaLog log4j plugin supports context preservation. It is possible to keep track or group logs based on the current user or session being used in a multi-threaded environment. To perform this type of logging one must use the [NDC](#) or [MDC](#) logging event. The following sets the OohLaLog token attribute on each log message being sent out.

Example:

```
org.apache.log4j.NDC.push(javax.servlet.http.HttpServletRequest.getRemoteUser());
```

Or

```
org.apache.log4j.MDC.put("token", javax.servlet.http.HttpServletRequest.getRemoteUser()); // you must use the MDC key "token"
```

Counters in Log4j

OohLaLog adds a new log level to log4j specifically for the purpose of incrementing counters and tracking metrics that may not reside within a specific log message. To increment a counter simply send a log message with the CountLevel and the counter will be named after the message passed.

```
import com.oohlalog.log4j.CountLevel;

...

Logger logger = Logger.getLogger(MyClass.class.getName());

logger.log(CountLevel.COUNT, "My Custom Counter"); // Message will become
the name of the counter
```

Your counters will appear in your dashboard charted by hour.

Grails

The Groovy/Grails plugin adds a log4j appender that forwards traffic to the OohLaLog API service. If you have set up your Grails application in OohLaLog and copied its API key, you are ready to follow these steps to begin sending your application log files to OohLaLog:

1. Modify `BuildConfig.groovy` repository and plug-ins sections.
2. Modify `Config.groovy`.

Add Oolalog to `BuildConfig.groovy`.

In your Groovy application's `buildConfig.groovy` file, add two lines. Add this line in the `repositories` section:

```
//Example line added to BuildConfig.groovy file, repositories section.  
  
repositories { mavenRepo  
  
"http://nexus.bertramlabs.com/content/repositories/publicReleases/" }
```

Add this line in the `plugins` section:

```
//Example line added to BuildConfig.groovy file, plug-ins section.  
  
plugins { compile  
' :ooh-la-log:0.4.7' }
```

Modify `Config.groovy`

Modify your `Config.groovy` file to add an appender to log4j that automatically forwards log messages of INFO level or higher to your OohLaLog application.

//Example adding an appender to log4j in `Config.groovy`.

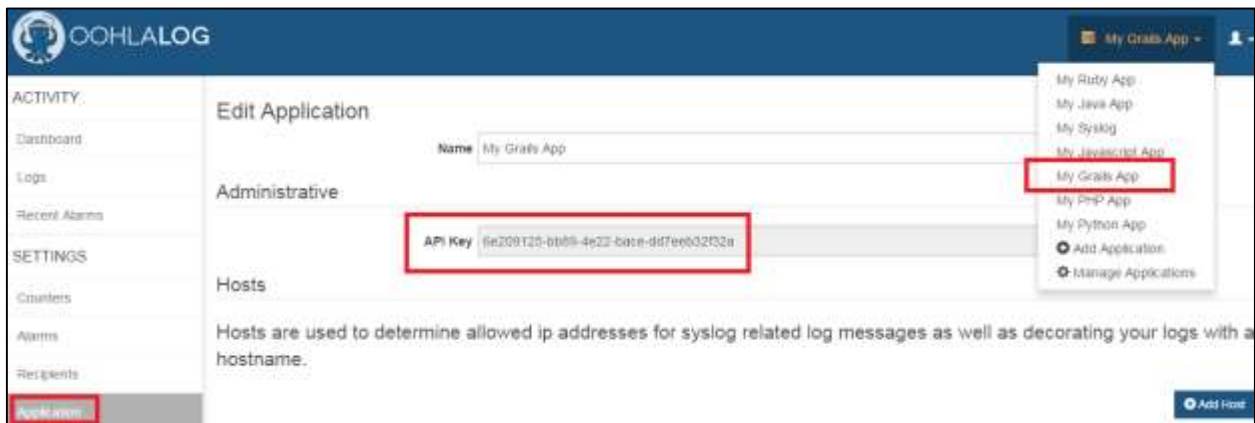
```
log4j = {  
    appenders {  
        appender new com.oohlalog.log4j.OohLaLogAppender( name: "oohlalog",  
authToken: "YOUR API KEY HERE")  
    }  
    root {  
        info 'oohlalog'  
    }  
}
```

Here are some appender properties you can set:

- host ["api.oohlalog.com"]
- path ["/api/logging/save.json"]
- port [80]
- secure [false]
- debug [false]
- hostName [null] // how the log source will appear



TIP: To find your Groovy/Grails application's API key, go to **Application** in the left-menu then use the pull-down menu in the upper right-hand corner to select your application, for instance "My Grails App." See the application's API key in the Administrative area.



Your Application's API Key

Verify Application

Substitute your OohLaLog's application's API key in the following code example to verify your application is submitting logs to OohLaLog.

```
//Example to verify if application runs correctly.  
  
/grails -Dreindex=1 run-app  
log4j = {  
  appenders {  
    appender new com.oohlalog.log4j.OohLaLogAppender( name: "oohlalog",  
authToken: "YOUR API KEY HERE")  
  }  
  root {  
    info 'oohlalog'  
  }  
}
```

Grails Artefact Logging and Counting

The following methods are available in all Domain Classes, Controllers, and Services.

```
void oohlaCount(String counterName, int increment = 1)

void oohlaLog(String level, String message, String category = "[artifact
name]",
              Object exceptionOrStringDetails = null, String token = null,
              Long timestamp = System.currentTimeMillis() )
```

These methods bypass log4j level threshold checks. Level should be one of the standard log4j levels, i.e. TRACE, DEBUG, INFO, WARN, ALL, ERROR, or FATAL.

Token can be used to relate a sequence of logs together (i.e. specific user activity).

See Also

For more information, see <http://groovy-lang.org/>

Next Step

After sending Grails logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).

.NET

Our OohLaLog adapter can be configured to send logs synchronously or asynchronously to the OohLaLog service. The OohLaLog adapter is used in conjunction with log4net, the Apache tool that sends log statements to a variety of output targets.

Before you begin you will need a .NET application set up in OohLaLog and its API key.

Usage

Whether you will be doing synchronous or asynchronous logging, include the OohLaLogAdapter library into your .NET project. The library DLL (OohLaLogAdapter.dll) can be found in the root of this repository:

https://github.com/oohlalog/oohlalog_log4net.

Asynchronous Logging

For asynchronous logging, modify the *log4net* configuration in your project configuration file to include the asynchronous OohLaLog appender:

```
<!-- This section contains asynchronous log4net configuration settings -->
<log4net>
  <appender name="AsyncOohLaLogAppender"
    type="OohLaLogAdapter.AsyncOohLaLogAppender, OohLaLogAdapter">
    <apikey value="YOUR API KEY HERE"/>
  </appender>
  <root>
    <level value = "ALL" />
    <appender-ref ref = "AsyncOohLaLoagAppender" />
  </ root>
</ log4net>
```

Synchronous Logging

To configure for synchronous logging, modify the *log4net* configuration in your project configuration file to include the synchronous OohLaLog appender:

```
<!-- This section contains synchronous log4net configuration settings -->
<log4net>
  <appender name="OohLaLogAppender"
type="OohLaLogAdapter.OohLaLogAppender, OohLaLogAdapter">
    <apikey value="YOUR API KEY HERE"/>
  </appender>
  <root>
    <level value = "ALL" />
    <appender-ref ref = "OohLaLoagAppender" />
  </ root>
</ log4net>
```



TIP: To find your .NET application’s API key, go to **Application** in the left-menu. From the pull-down menu in the upper right-hand corner select your application. “My .NET App” is shown in the example. See the application’s API key in the Administrative area.



Your Application’s API Key

See Also

For more information, see the .NET documentation at: <http://msdn.microsoft.com/en-us/library/w0x726c2%28v=vs.110%29.aspx>

Next Step

After sending .NET logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).

PHP

The PHP connector removes the default debugging logic and replaces it with a logger to OohLaLog. You only need to set up an API key to be good to go.

Before you begin you will need a PHP application set up in OohLaLog and the API key created for this application.

First you will assign values to global variables then add the OohLaLog API key to every PHP script or add it globally.

Add Values to Global Variables

This plugin is configured by assigning a these values to global variables within your PHP application.

- `oohLaLogApiKey` – Your OohLaLog Python application’s API key.
- `OLL_LOG_FILE` - (optional) Save log messages to a file that defaults to `/usr/local/php/error.log`.
- `OLL_PRINT_ERRORS` - (optional) Display OohLaLog plugin related errors (default `true`).

Add OohLaLog API Key

At the top of each PHP script in your application, or in a file that is included everywhere, add the following:

```
//Example adding OohLaLog API key.
```

```
pip install Oolalog
```

```
<?php
    $OLL_LOG_LEVEL = 7;
    $oohLaLogApiKey = 'YOUR API KEY HERE';
    $oohLaLogEnabled = true;
    require('oohLaLogger.php');
    trigger_error("log undefined", E_USER_ERROR);
    phpinfo();
?>
```

Anything that happens before the `require` will not be sent to OohLaLog.

Limitations

This plugin currently uses the `exec` command to fork a curl process. You won't get a response but it no longer blocks in PHP while running.



TIP: To find your PHP application's API key, go to **Application** in the left-menu. Use the pull-down menu in the upper-right corner to select your application, for instance "My PHP App." The application's API key is in the Administrative area.



Your Application's API Key

See Also

For more information, see <http://php.net/manual/en/langref.php>.

Next Step

After sending your application logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).

Python

The Python OohLaLog plugin adds a standard Python log appender to your application's Python logger. This allows OohLaLog to graphically view Python logs in a non-blocking, real-time, easy way.

Before you begin you will need a Python application set up in OohLaLog and the API key created for this application. Next you will install the OohLaLog plugin using *pip* then add your OLL API key into your application.

This section also includes optional steps for verifying log forwarding, configuring the OLL appender, and picking the date/time of the logs to be visually graphed.

Pypi Plug-in Install

Use *pip* to install the plugin.

```
//Example installing OohLaLog.
```

```
pip install Oohlalog
```

Add API Key

Adding your API key into your application varies slightly depending on is you are using Python 2.7 or 3.0:

Python 2.7

```
//Example adding OohLaLog API key for Python 2.7.
```

```
import logging
from oohlalog import logger

logs = logging.getLogger('test')
logs.addHandler(logger.OohLaLogHandler('YOUR OLL API KEY HERE'))
```

Python 3.0

```
//Example adding OohLaLog API key for Python 3.0.
```

```
import logging
from py3 import logger

logs = logging.getLogger('test')
logs.addHandler(logger.OohLaLogHandler('YOUR OLL API KEY HERE'))
```



TIP: To find your Python application's API key, go to **Application** in the left-menu then use the pull-down menu in the upper right-hand corner to select your application, for instance "My Python App." See the application's API key in the Administrative area.



Your Application's API Key

(Optional) Verify Log Forwarding.

Now go check out the OLL logs page or dashboard for your application to see your logs. Here are sample log calls that will be forwarded to OohLaLog. They are the same as logging you already have.

//Example logs forwarded.

```
logs.info('test info')
logs.debug('test debug')
logs.warning('test warning')
logs.error('test error')
logs.critical('test critical')
```

(Optional) Configure OLL appender

While OohLaLog appender supports these configurations, only `apiKey` is required:

- `apiKey` - (required) OohLaLog API Key
- `threshold` - number of logs before sending to OohLaLog
- `timeout` - number of seconds to keep logs before sending to OohLaLog

NOTE: `timeout` overrides `threshold`.

- `formatter` - `logging.Formatter`

(<http://docs.python.org/2/library/logging.html#logging.Formatter>) overrides the default detail string sent to OohLaLog

```
//Example OLL appender configuration.
```

```
OohLaLogHandler(apiKey, threshold=100, timeout=5, formatter=None)
```

See Also

For extensive Python documentation, see <https://www.python.org/doc/>

Next Step

After sending your application logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).

Javascript

OohLaLog provides an interface to push logs and increment counters directly from your Javascript code. You can also override your console object to push console calls to OohLaLog. Simply add the `olo.js` file found [here](#) to the end of your web application.

Before you begin you will need a Javascript application set up in OohLaLog and the API key created for this application.

Usage

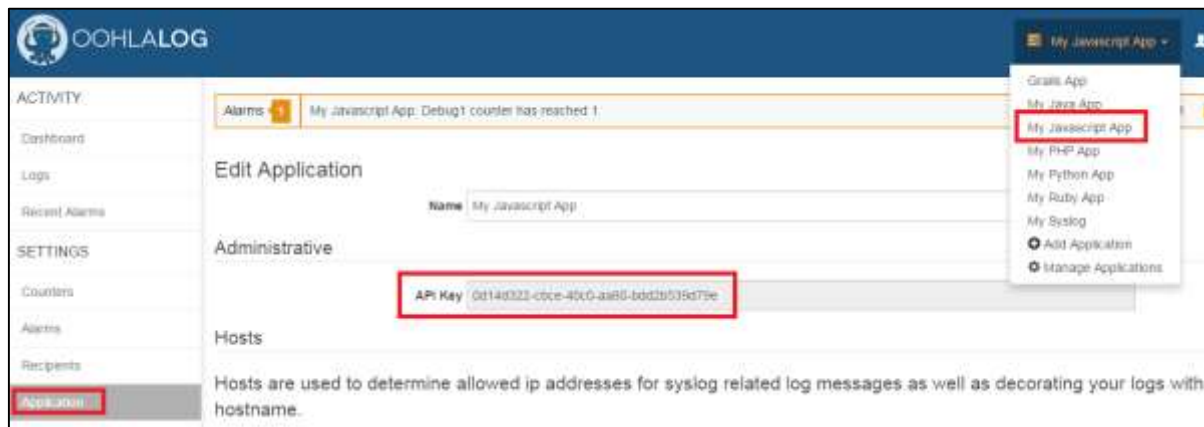
Copy and paste the following code to the end of your application. Be sure to add your OohLaLog API key. It's that simple.

```
olo.count(code,incr,name+,callback+,YOUR_API_KEY_HERE+)  
olo.info(msg,category,details+,priority+,token+,callback+,YOUR_API_KEY_HERE +)  
olo.warn(msg,category,details+,priority+,token+,callback+,YOUR_API_KEY_HERE +)  
olo.error(msg,category,details+,priority+,token+,callback+,YOUR_API_KEY_HERE +)  
olo.debug(msg,category,details+,priority+,token+,callback+,YOUR_API_KEY_HERE +)  
olo.trace(msg,category,details+,priority+,token+,callback+,YOUR_API_KEY_HERE +)
```

Now you can begin logging.



TIP: To find your Javascript application's API key, go to **Application** in the left-menu. Use the pull-down menu in the upper right-hand corner to select your application, for instance "My Javascript App." See the application's API key in the Administrative area.



Your Application's API Key

Low Level APIs.

This code implements a low-level Javascript API. Examples follow.

```
olo.counter.get(cfg)
olo.counter.increment(cfg)
olo.counter.reset(cfg)
olo.logger.get(cfg)
olo.logger.save(cfg)
olo.logger.delete(cfg)

// Examples of low level API calls

olo.setApiKey('YOUR_API_KEY_HERE');

olo.counter.increment({code: 'TEST-CT-1', callback: function(obj)
{alert('success='+obj.success)}});
olo.counter.get({code: 'TEST-CT-1', callback: function(obj)
{alert('count='+obj.data.count)}});
olo.warn('my msg', 'my cat');
olo.logger.save({category: 'spanky', level: 'DEBUG', message: 'Howdy', callback:
function(obj) {alert('success='+obj.success)}});
olo.trace('my msg', 'my cat');
```

NOTE: All methods support chaining.

See Also

For more information, see Javascript documentation at <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Next Step

After sending your application logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).

OracleDB

TBD

Syslog

OohLaLog supports listening for your log messages via the *syslog* protocol. This allows IT operations and other non-supported platforms to use the logging interface. Aggregate your logs by setting up your *rsyslog.conf* then point your logs to OohLaLog.

Rsyslog is an open-source software utility for forwarding log messages in an IP network. The official website defines the utility as "the **rocket-fast system for log processing.**" It implements the basic *syslog* protocol, and extends it with content-based filtering, rich filtering capabilities, flexible configuration options and adds features such as using TCP for transport.

Configuration

Go to the server you wish to forward your logs from and add a rule in your *rsyslog.conf* file. This rule allows the OohLaLog platform to monitor for *syslogs* at `syslog.oohlalog.com` listening for UDP on port **8514** and TCP on **6514**.

```
// Add OohLaLog rule in rsyslog.conf.
```

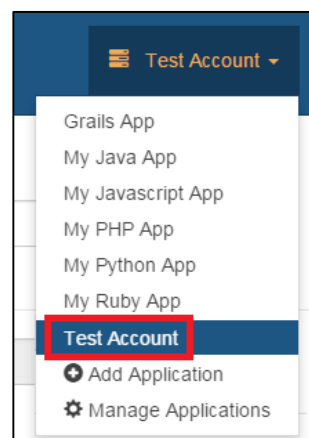
```
# UDP Example
```

```
*.* @syslog.oohlalog.com:8514
```

```
#TCP Example
```

```
*.* @@syslog.oohlalog.com:6514
```

Next, select **Applications** from the side bar menu then your OohLaLog application from the drop-down box (upper right-hand corner).



Select Your Application

Add the hostnames and the IP addresses of the servers you wish to accept logs from.

The screenshot shows the OohLaLog web interface. The top navigation bar includes the OohLaLog logo and a user profile for 'Test Account'. The left sidebar is divided into 'ACTIVITY' (Dashboard, Logs, Recent Alarms) and 'SETTINGS' (Counters, Alarms, Recipients, Application, Documentation, Community). The 'Application' setting is selected, showing an 'Edit Application' form with fields for 'Name' (Test Account) and 'API Key' (3f308a8-014b-427a-9754-ad3a25016440). Below this is the 'Hosts' section, which explains that hosts determine allowed IP addresses for syslog messages. A table lists a host with 'Hostname' 'MyServer' and 'IP Address' '1.2.3.4'. There are 'Add Host' and 'Remove' buttons.

Enter Hostname and IP Address

Once this information is saved, select Dashboard. Within a few minutes logs should start appearing for your application.

See Also

For more information, see the fine rsyslog documentation at http://www.rsyslog.com/doc/rsyslog_conf.html

Next Step

After sending logs to OohLaLog, the next step is to setup a counter, an alarm, and a recipients who will receive the email or text messages when the alarm threshold is reached. See [Create Counters, Recipients and Alarm Notifications](#).