



# Motherload for osTicket

by **Cartmega**

05/06/2022

Aristotelis Joannou



## Table of Contents

INTRODUCTION	3
ONLINE DEMO	4
INSTALLATION	5
TESTING	9
Client & Agent Side Demo	9
PLUGIN PAGE CONFIGURATION	10
Notification area	10
Allowed Signals	10
Enabled Plugins	10
Valid Plugins	11
Debugger	11
Disable Plugin Syntax Checking	12
CUSTOM PLUGIN DEVELOPMENT	13
Location	13
Naming convention	13
Compulsory variables & functions	13
Plugin example (plugin_test.php)	14
Motherload Capabilities	16
Functions	16
Variables	17
jQuery / Javascript	18
TOPICLISTS	19
SUPERLISTS	20
LICENSE VERIFICATION	21
SOLUTIONS TO PROBLEMS	22

**The Ultimate in**  
Rapid Plugin Development

Your custom plugins  
up and running in under 5 minutes  
on the event you specify!

► for osTicket ◀  
► saves a ton of time ◀



**MOTHERLOAD**  
plugin controller  
for osTicket

**RUN YOUR CUSTOM CODE  
ON THE EVENT YOU WANT  
AND EVEN DEBUG WHEN LIVE!**

**'Don't write a new plugin...  
Plug into motherload!'**

*'Plugin Creation & Handling  
with debugging plus  
customer friendly  
on production servers!'*



## INTRODUCTION

[Motherload](#) is a plugin controller that allows rapid plugin development in osTicket. Have your custom plugins up and running in under 5 minutes, to handle any osTicket event.

Motherload makes osTicket custom plugin development easy, fast, enjoyable and bomb proof!

At the time when an osTicket event is fired a signal is sent out which is caught by Motherload if configured to do so on the plugin configuration page. It then checks to see if you have any custom plugins available and enabled to handle this particular signal. If so, then Motherload sends all signal related information to your custom plugin and allows it to run.

Runtime errors will appear in the osTicket syslog without bothering your clients.

- Requires PHP version 7.2.x or higher. Proper operation cannot be guaranteed with earlier versions.
- No core file modifications necessary!



## ONLINE DEMO

Did you know that there is an online demo of various Motherload plugins?

If you have not had a chance to try Motherload, you can check it out right now!

Of course Motherload is the plugin controller running behind the scenes in these demos. Nevertheless these plugins share and exhibit all Motherload features and are prime examples of what can be achieved.

When you click the button below you will be taken to the [Cartmega demo website](#), where you can test drive Motherload plugins like Topiclists, Superlists and License Verification.





# INSTALLATION



**MOTHERLOAD**  
plugin controller  
for osTicket

RUN YOUR CUSTOM CODE  
ON THE EVENT YOU WANT  
AND EVEN DEBUG WHEN LIVE!

© 2019 WWW.CARTMEGA.COM



# cartmega

INSTALLATION  
CONFIGURATION  
TESTING

With Motherload, you can rapidly develop  
osTicket plugins in minutes instead of hours!

A revolutionary solution for osTicket developers.

[www.cartmega.com](http://www.cartmega.com)



1. Motherload is available online here:


<https://www.cartmega.com/motherload-for-osticket.html>

2. Download & Unzip.

- a. Download our plugin.  
(motherload\_for\_osTicket\_v.x.x.x.zip)

☐ Name

Date modified

 motherload\_for\_osTicket\_v.1.4.3.zip

22/05/2022 16:15

3. Upload Motherload.

- a. Extract the contents of motherload\_for\_osTicket\_v.x.x.x.zip,  
onto your hard drive.

- b. Open the folder 'UPLOAD'  
where you just extracted the  
zip file.

- c. FTP into your osTicket  
installation & find the osTicket  
root directory.

- d. Select all files & folders within  
the 'UPLOAD' folder (on your  
computer) and upload all to the  
osTicket root directory of your  
FTP server.

☐ Name

Date modified

Type

 include

07/02/2022 03:58

File folder

 scp

07/02/2022 03:58

File folder

Name

Size

Changed

Rights

 api

26/04/2022 19:26:51

rw-xr-xr-x

 apps

21/04/2022 15:52:54

rw-xr-xr-x

 assets


21/04/2022 15:52:54

rw-xr-xr-x

 css


21/04/2022 15:52:54

rw-xr-xr-x

 images

21/04/2022 15:52:54

rw-xr-xr-x

 include

28/04/2022 00:28:36

rw-xr-xr-x

 js

21/04/2022 15:52:57

rw-xr-xr-x

 kb

21/04/2022 15:52:57

rw-xr-xr-x

 pages


21/04/2022 15:52:57

rw-xr-xr-x

 scp

25/05/2022 02:25:49


rw-xr-xr-x

 account.php

6 KB

20/04/2022 14:21:06


rw-r--r--

 ajax.php

2 KB

20/04/2022 14:21:06


rw-r--r--

 avatar.php

2 KB

20/04/2022 14:21:06


rw-r--r--

 bootstrap.php

16 KB

19/05/2022 20:04:18

rw-r--r--

 captcha.php

1 KB

20/04/2022 14:21:06

rw-r--r--

## Product Download

DOWNLOADS (9)

Motherload for osTicket

1.12.x » 1.16.x

Version: 1.4.4

Updated: 04/06/2022



#### 4. Install Motherload in osTicket.

- Login into osTicket.
- Go to Admin panel > Manage > Plugins.
- Click 'Add New Plugin' & click [Install] next to 'osTicket Motherload'.
- To enable the plugin, place a check mark next to 'osTicket Motherload'. Click the 'More' button and click [Enable].

## Currently Installed Plugins

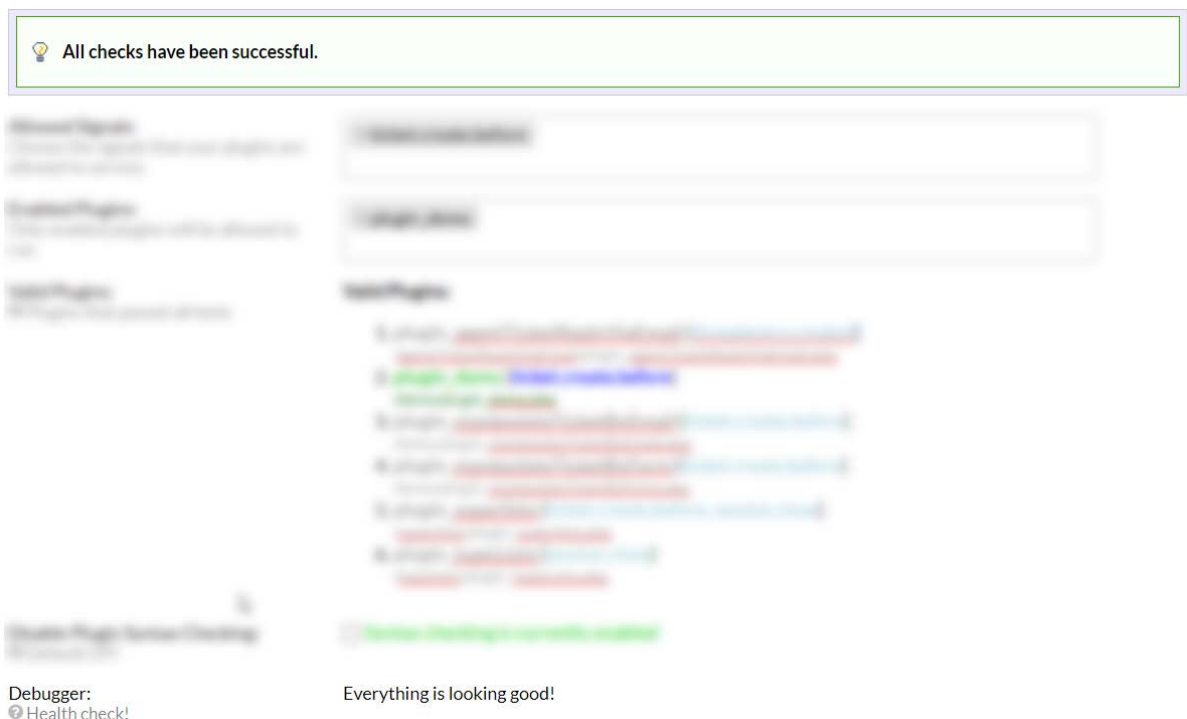
Add New Plugin
More

	Plugin Name	Version	Status	Date Installed	
<input checked="" type="checkbox"/>	osTicket MotherLoad	1.4.4	Disabled	6/4/22	<div> Enable Disable Delete </div>
<input type="checkbox"/>	Real Dynamic Lists	1.1.2	Enabled	4/28/22	
<input type="checkbox"/>	Real Dynamic Lists MS SQL	1.1.3	Enabled	4/28/22 3:09 AM	
<input type="checkbox"/>	Rejection Notifier	1.2.2	Disabled	4/28/22 4:25 PM	

Select: All None Toggle


Page: 1

- e. Click on 'osTicket Motherload' to enter the plugin settings. Just make sure that there are no errors reported.



Motherload requires no customization. All configuration is done on the plugin page. Error reporting is done on the plugin page but also in the osTicket syslog.

5. Install the demo plugin (plugin\_demo.php).
  - a. Login into osTicket.
  - b. Go to Admin panel > Manage > Plugins > osTicket MotherLoad.
  - c. In Allowed Signals: add 'ticket.create.before'.
  - d. In Enabled Plugins: add 'plugin\_demo'.
  - e. Click [Save]. You should see no errors being reported in the debugger.

 All checks have been successful.

Allowed Signals:  
Choose the signals that your plugins are allowed to service.

✕ ticket.create.before

Enabled Plugins:  
Only enabled plugins will be allowed to run.

✕ plugin\_demo

Valid Plugins:  
🔍 Plugins that passed all tests

Valid Plugins:

1. plugin\_agentTicketReplyViaEmail [threadentry.created]  
/agentTicketReplyViaEmail/plugin\_agentTicketReplyViaEmail.php

2. plugin\_demo [ticket.create.before]  
/demo/plugin\_demo.php

3. plugin\_manipulateTicketByEmail [ticket.create.before]  
/demo/plugin\_manipulateTicketByEmail.php

4. plugin\_manipulateTicketByForm [ticket.create.before]  
/demo/plugin\_manipulateTicketByForm.php

5. plugin\_superlists [ticket.create.before, session.close]  
/superlists/plugin\_superlists.php

6. plugin\_topiclists [session.close]  
/topiclists/plugin\_topiclists.php

Disable Plugin Syntax Checking:  
🔍 Default: OFF

☐ Syntax checking is currently enabled

Debugger:  
🔍 Health check!

Everything is looking good!

Congratulations. You have completed the installation of Motherload.

You can also uninstall it by going to Admin panel > Manage > Plugins, place a check mark next to 'osTicket Motherload', click the 'More' button and click [Delete].

If you want to disable a plugin, just remove it from the list of enabled plugins by clicking the [x] next to the plugin tag and then clicking [Save]. If you remove a plugin it is a good idea to also remove the associated signal from the 'Allowed Signals' list, if it is not being used by other enabled plugins.

You can now proceed to test the Motherload demo.



# TESTING

## Client & Agent Side Demo


Visit the client side of your osTicket installation and click 'Open a New Ticket'. Scroll to the bottom of the page and click 'Create Ticket'.


This is a demo message from dbgecho.  
dbgecho is useful when you are forced to debug your plugin in a live environment without any clients seeing your debug messages.  
To prevent this message showing on the client screen, provide your IP address in the \$debug\_ip array.

**SUPPORT CENTER**  
Support Ticket System

Guest User | [Sign In](#)

[Support Center Home](#) [Knowledgebase](#) [Open a New Ticket](#) [Check Ticket Status](#)

 Your email is not "support@cartmega.com"! Please try again.

 Unable to create a ticket The email you have provided seems invalid.

**Open a New Ticket**

Please fill in the following information:

**Contact Information**


**Email Address**

Email Address is a required field

**Full Name \***

Full Name is a required field

**Phone Number**  **Ext:**

**Warning:** Your email is not "support@cartmega.com"! Please try again. 

You should see a modal pop-up alerting you to the fact that the email address you entered does not qualify for the creation of a new ticket. For the ticket to be created successfully you should enter 'support@cartmega.com' in the email field.


When you click 'OK' to dismiss the warning, you should also see the same warning message in a yellow rectangle just above the form title. Just below it there should be an error message in a similar red rectangle. Finally, at the very top of the page there should be a plain text message from dbgecho (debug echo message).

The same exact things should happen on the agent side of osTicket.

# PLUGIN PAGE CONFIGURATION

The plugin configuration page includes the following:

## Notification area

 All checks have been successful.

Motherload will report its own messages here.

## Allowed Signals

Allowed Signals:  
Choose the signals that your plugins are allowed to service.

× ticket.create.before

Choose the signals that your plugins are allowed to service. If your plugin requires signals 'session.close' and 'ticket.create.before' to function, then you should select both of these plugins to appear in this list.

Motherload will subscribe to all these signals in osTicket and if a signal is triggered, it will pass handling over to the plugins enabled to handle it.

## Enabled Plugins

Enabled Plugins:  
Only enabled plugins will be allowed to run.

× plugin\_demo

Choose the plugins you want to enable. Each plugin is associated with one or more signals. If the plugin is enabled as well as its associated signals, then the plugin will be able to handle the signal when that is triggered.

**NOTE:** The lists 'Allowed Signals' as well as 'Enabled Plugins' support auto-complete for fast searching within the lists. After clicking within the list, just start typing.

× ticket.create.before

sess

session.close

× plugin\_demo

**NOTE:** If you want to disable a signal or plugin, just remove it from the list by clicking the [x] next to its tag. If you remove a plugin it is a good idea to also remove the associated signal from the 'Allowed Signals' list, if it is not being used by other enabled plugins.

## Valid Plugins

Valid Plugins:

🔍 Plugins that passed all tests

Valid Plugins:

1. `plugin_agentTicketReplyViaEmail` [`threadentry.created`]  
/agentTicketReplyViaEmail/plugin\_agentTicketReplyViaEmail.php
2. `plugin_demo` [`ticket.create.before`]  
/demo/plugin\_demo.php
3. `plugin_manipulateTicketByEmail` [`ticket.create.before`]  
/demo/plugin\_manipulateTicketByEmail.php
4. `plugin_manipulateTicketByForm` [`ticket.create.before`]  
/demo/plugin\_manipulateTicketByForm.php
5. `plugin_superlists` [`ticket.create.before, session.close`]  
/superlists/plugin\_superlists.php
6. `plugin_topicLists` [`session.close`]  
/topiclists/plugin\_topicLists.php

In this section Motherload will present a list of recognized plugins inside the /include/plugins/motherload/plugins/ folder along with some useful information.

It is important to note that the signals required by each plugin are reported next to the plugin name as shown below.

1. `plugin_agentTicketReplyViaEmail` [`threadentry.created`]

When you enable a plugin, it will appear in green.

1. `plugin_agentTicketReplyViaEmail` [`threadentry.created`]

**NOTE:** Just because a plugin appears enabled does not mean that it will also be functional. Do not forget to also enable any signals associated with it.

## Debugger

Debugger:

🔍 Health check!

Problems were found:

1. Error::  
ML::motherload - Plugin class invalid: Plugin will not be loaded  
s/demo/plugin\_manipulateTichketByEmail.php' /include/plugins/motherload/plugin  
The class found 'plugin\_manipulateTicketByEmail' was unexpected. Was expecting  
'plugin\_manipulateTichketByEmail'

Here Motherload will report any issues after checking the accessibility and executability of any available plugins.

Any errors must be fixed before a plugin will become available to be enabled.

## Disable Plugin Syntax Checking

Disable Plugin Syntax Checking:  
🔍 Default: OFF

☒ Syntax checking is currently disabled (most errors will still be reported in the debugger below)

Motherload performs syntax checking on your plugins by making use of the PHP '**exec**' command. In certain server setups this functionality is disabled, thus causing the following error message immediately after the installation of Motherload:

Debugger:  
🔍 Health check!

Problems were found:

```
1. Error::
ML::motherload - Plugin syntax error: Plugin
'/include/plugins/motherload/plugins/demo/plugin_demo.php' failed syntax checking and was not
loaded.

Array
(
    [0] => Could not open input file:
'/include/plugins/motherload/plugins/demo/plugin_demo.php'
)
```

To overcome the inability of the server to allow this functionality you can disable syntax checking in Motherload by ticking the check box and saving.

When disabled, Motherload will not conduct syntax checking on your plugin scripts before trying to include and then execute them.

This should not be a major setback since most errors are also caught and reported in the debugger when Motherload later attempts to include the script.

Disabling plugin syntax checking is useful mostly for shared hosting or other installations where PHP suffers CLI execution limitations.

If the debugger below is showing '**Plugin syntax error: [0] => Could not open input file: path-to-script.php**', then you can bypass syntax checking by checking this option.

You can also confirm the problem programmatically by executing **php -l path-to-script.php** which should return '**Could not open input file: path-to-script.php**'.

# CUSTOM PLUGIN DEVELOPMENT

Motherload is a plugin controller designed for the specific purpose of assisting in the rapid application development of plugins for the osTicket platform.

In order for us to develop Motherload plugins we should keep the following in mind:

## Location

All plugin files must be placed in their own folder in the following path:

`INCLUDE_DIR/plugins/motherload/plugins/`

Plugins placed in other folders will not be processed!

Follow the naming convention below to avoid clashes.

**NOTE:** Each folder may contain any number of plugins.

## Naming convention

**Folder name** Can be anything, but to avoid confusion you may consider using the same name as the name of your plugin (`pluginname`). The folder name may contain spaces but your `pluginname` cannot.

**File name** `plugin_pluginname.php` (no spaces allowed)

**Class name** `class pluginname extends motherloadPlugin {}`

## Compulsory variables & functions

Inside your plugin class you must have the following:

```
const signalConnect = '{enter here the signal you want}';  
public function run() {enter here what you want your plugin to do}  
protected $debug_ip = {true or false or array(xxx.xxx.xxx.xxx)}; (optional)
```

## Plugin example (plugin\_test.php)

Open folder `/include/plugins/motherload/plugins` and create a new folder 'test'.

Open the folder you just created and create a file called `'plugin_test.php'`.

Open the file and paste the following code (without the line numbers) and then save.

```
0. <?php
1. class plugin_test extends motherloadPlugin {
2.     const signalConnect = 'ticket.create.before';

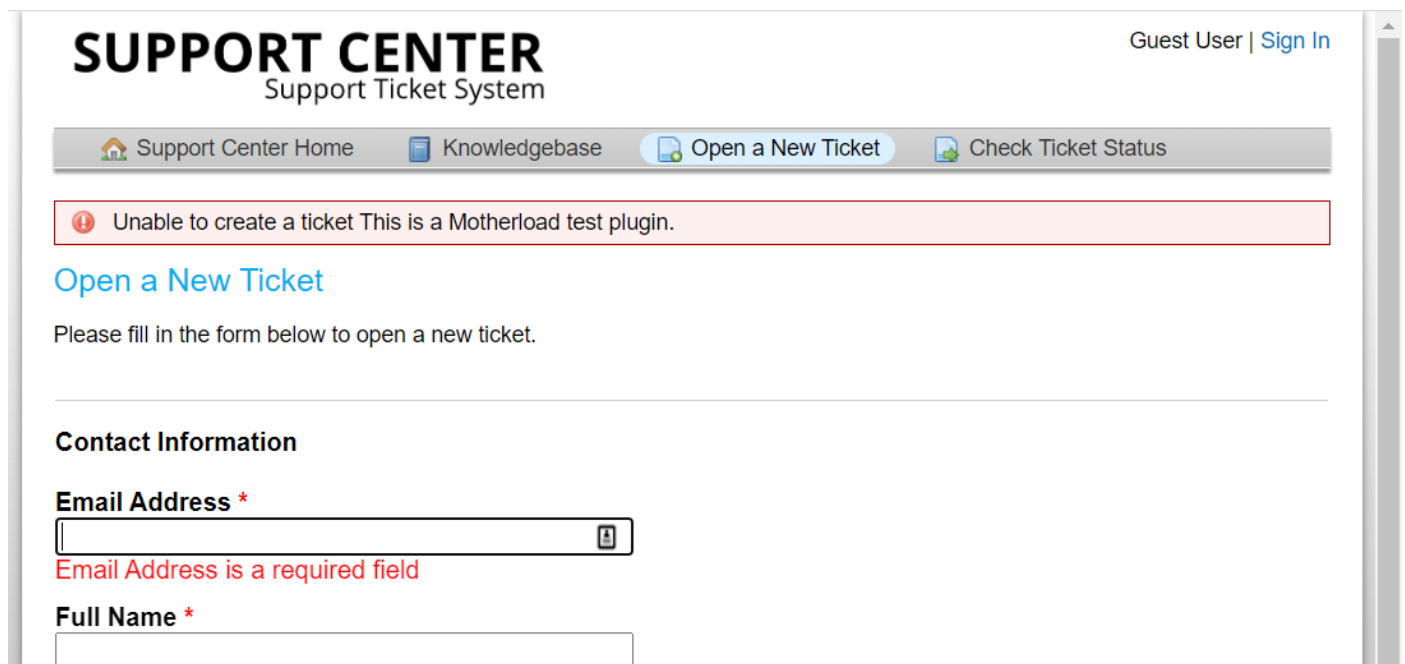
3.     public function run() {

4.         $this->addErr('Ticket failed.', 'This is a Motherload test plugin. ');
5.         return false;
6.     }
7. }
```

Navigate to the Motherload plugin page in osTicket and enable the plugin as well as the signal it requires (`'ticket.create.before'`) and click [Save Changes].

Visit the client side of your osTicket installation and click 'Open a New Ticket'. Scroll to the bottom of the page and click 'Create Ticket'.

You should see the following:



The screenshot shows the 'SUPPORT CENTER' header with the subtitle 'Support Ticket System'. In the top right corner, it says 'Guest User | Sign In'. Below the header is a navigation bar with links: 'Support Center Home', 'Knowledgebase', 'Open a New Ticket' (highlighted), and 'Check Ticket Status'. A red error message box states: 'Unable to create a ticket This is a Motherload test plugin.' Below this is a section titled 'Open a New Ticket' with the instruction 'Please fill in the form below to open a new ticket.' The form has a section for 'Contact Information' with two fields: 'Email Address \*' and 'Full Name \*'. The 'Email Address \*' field is empty and has a red error message below it: 'Email Address is a required field'.

This simple plugin takes under 2 minutes to create and immediately can interact with the osTicket system. You simply cannot beat that kind of performance!



**What our test plugin does:**

1. Here we define that the code that follows is a plugin for motherload. Our plugin class has a name which connects it to the filename our code is in.
2. The signalConnect constant informs Motherload that our plugin wants to run whenever the 'ticket.create.before' signal is triggered.
3. The run() function is what Motherload will call when the signal is triggered, so we will place our code in here.
4. Here we add an error to osTicket's error array with the purpose of displaying it to the user on the new ticket creation page.
5. We exit our function by returning a false value, thus informing osTicket that something went wrong with the ticket creation process. osTicket will therefore check its error array for any messages to show to the user and will find the one we added there previously.

Sky is the limit with what you can do with Motherload.

osTicket plugin development has just become child's play.

No more having to design an interface and deal with all the intricacies of plugin design. The developer can therefore concentrate on writing code.

For those plugins which do not require a user interface they can be implemented in minutes in pure PHP and with a few variables for configuration.

## Motherload Capabilities

In the previous section we saw how we can create a Motherload plugin. What capabilities does Motherload provide for use in our plugins?

### Functions

#### To be used inside the class definition:

```
protected $debug_ip = true; // true or false or array('xxx.xxx.xxx.xxx')
```

The `$debug_ip` value controls who can see the `dbgecho` messages.

- True (everyone can see)
- False (no one can see)
- array('xxx.xxx.xxx.xxx', 'xxx.xxx.xxx.xxx') (IP restricted)

#### To be used inside the run() function:

```
$this->dbgecho('message')
```

Useful when you are forced to debug your plugin in a live environment without any clients seeing your debug messages.

```
$this->ostlog(LOG_WARN, 'Title', 'Message');
```

You can easily write to the osTicket syslog.

```
$this->addError('Title', 'Message');
```

Add an osTicket error to its error array. The errors are shown by osTicket near the top of the page only if you disrupt a process like ticket. e.g. when you attach your plugin to ticket.create.before, you can use the following code to induce an error in osTicket.

```
if ($_POST) {  
    $this->addError('Title', 'Message');  
    return false;  
}
```

```
$this->addAlert('Title', 'Message'); // use addDialog for pop-up only
```

Show a modal pop-up alert to the user and a warning on the page.

```
$this->dbgecho('<pre>'. print_r($this->ml, 2). '</pre>');
```

Show all data in the Motherload object.

## Variables

### Motherload variables:

The variables below are always available to any Motherload plugin.

Variable	Description
<code>\$this-&gt;ml</code>	<p>When a signal is triggered, osTicket sends the <b>object</b> involved and possibly some <b>data</b> involved in the event.</p> <pre><code>\$this-&gt;ml-&gt;signal [ ' object' ]</code></pre> <pre><code>\$this-&gt;ml-&gt;signal [ ' data' ]</code></pre> <p>An explanation of the above objects is beyond the scope of this guide. <a href="#">...more information on plugin development</a>.</p> <p>Motherload also enriches this object with:</p> <pre><code>\$this-&gt;ml-&gt;cdata (contains data from posted fields)</code></pre> <pre><code>\$this-&gt;ml-&gt;_POST (contains POST information)</code></pre>
<code>\$this-&gt;parent</code>	access to all motherload objects (the parent of our plugin)
<code>\$this-&gt;plugininfo</code>	access to plugin info
<code>\$this-&gt;url</code>	access to current URL information
<code>\$this-&gt;errors</code>	access to the Motherload error array
<code>\$this-&gt;signalConnect</code>	contains the actual signal name which was just triggered

### osTicket variables:

Since we are working in osTicket, we may also be interested in its own variables.

To find out what osTicket global variables are available to work with, use:

```
$this->dbgecho(' <pre>' . print_r(array_keys($GLOBALS), 2) . ' </pre>');
```

Variable	Description
<code>\$ost</code>	access to osTicket information & logging
<code>\$chg</code>	access to osTicket configuration
<code>\$_db</code>	access to the active mysqli connection
<code>\$thisclient</code>	access to the osTicket logged-in client object
<code>\$thisstaff</code>	access to the osTicket logged-in staff object

In order to work with an osTicket global variable we have to first import it into our own **run()** function like this:

```
global $ost, $cfg, $thisclient, $errors;
```

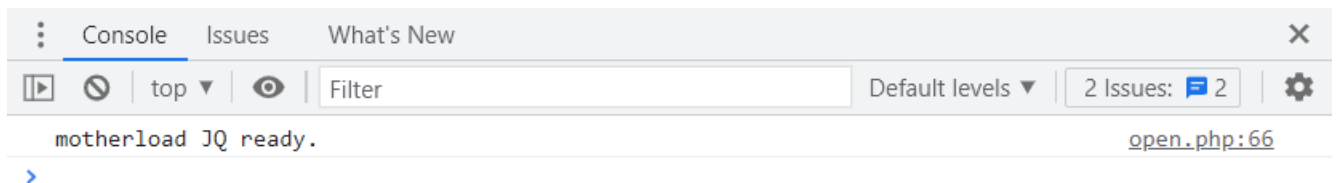
Furthermore, in order to work with some osTicket objects we have to remember to include necessary files before our plugin `class()` declaration. e.g.

```
include_once(INCLUDE_DIR . 'class.ticket.php');
require_once(INCLUDE_DIR . 'class.task.php');
include_once INCLUDE_DIR . 'class.thread_actions.php';
require_once(INCLUDE_DIR . 'class.staff.php');
```

You can therefore customize the behavior of your Motherload plugin based on URL, client, staff and much more! This means your options are endless.

## JQuery / Javascript

When Motherload is triggered to run one of its plugins, it will also attempt to inject some Javascript into the web page involved. The reason is in order to support commands such as `$this->addAlert` and `$this->addDialog` which are made available to all plugins.



When Motherload fully loads in the browser it outputs the above message in the developer's console and can be seen by pressing **F12**.

Any errors regarding this client side functionality will also be displayed here.

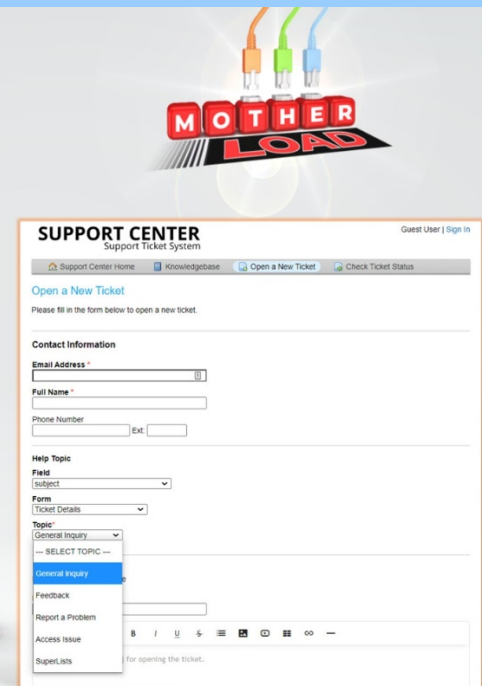
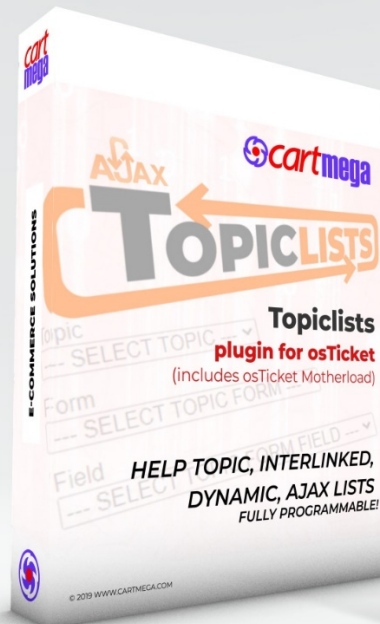
Of course your own plugin can also inject Javascript into the web page. All you have to do is `echo` a `<script>` section from within your `run()` function.

```
class plugin_test extends motherloadPlugin {
    const signalConnect = 'ticket.create.before';
    public function run() {
        $this->addErr('Unable to create a ticket', 'This is a Motherload
        test plugin. ');
        echo ' <script>alert("hello world")</script>';
        return false;
    }
}
```

**The Ultimate in**  
Help Topic Categorization  
**Topiclists can be**  
**interlinked & nested**  
**& populated with AJAX**  
**and even tailored to individual clients!**

► **for osTicket** ◀  
► **includes motherload** ◀

*'It is finally here...  
the holy grail  
in the ticket creation process!'*



## TOPICLISTS

Enhance your Help Topics with multiple, custom, smart, dynamic, interlinked and nested drop-down lists which are updated with ajax from any database.

Check out the video and online demo below:



**cartmega demo**

**The Ultimate in**  
Smart Dynamic Lists

Superlists can be  
interlinked & grouped,  
populated with AJAX  
and even tailored to individual clients!

► for osTicket ◀  
► includes motherload ◀

'It is finally here...  
**the holy grail**  
in the ticket creation process!'

**SUPERLISTS**  
plugin for osTicket  
(includes osTicket Motherload)

INTERLINKED, GROUPED,  
DYNAMIC, AJAX LISTS  
FULLY PROGRAMMABLE!

**SUPPORT CENTER**  
Support Ticket System

Open a New Ticket

Please fill in the form below to open a new ticket.

Contact Information

Email Address

Full Name

Phone Number  
 Ext.

Help Topic  
SuperLists

SuperLists Form

Topic  
SuperLists

Form  
SuperLists Form

Field  
SELECT TOPIC FORM FIELD

sup\_Topic  
sup\_Form  
sup\_Field

## SUPERLISTS

What Topiclists can do for Help Topics, Superlists can do for Forms.

Enhance your forms with multiple, custom, smart, dynamic, interlinked and nested drop-down lists which are updated with ajax from any database. The user selections are saved in their ticket.

Check out the video and online demo below:

**SUPERLISTS**  
plugin for osTicket  
(includes osTicket Motherload)

INTERLINKED, GROUPED,  
DYNAMIC, AJAX LISTS  
FULLY PROGRAMMABLE!

**INSTALLATION  
CONFIGURATION  
TESTING**

Visit [oscdemo.cartmega.com](http://oscdemo.cartmega.com) for a test drive.

[www.cartmega.com](http://www.cartmega.com)

**cartmega demo**





**The Ultimate in License Verification**

Checks client supplied order info against multiple sales channels local and online!

- for osTicket ◀
- fully expandable ◀ easily add custom channel plugins!
- includes motherload ◀

*'If you sell digital products online you need this!'*

**License Verification plugin for osTicket**  
(includes osTicket Motherload)

**VERIFY CLIENT'S LICENSE BEFORE PROVIDING SUPPORT**

**SUPPORT CENTER**  
Support Ticket System

Support Center Home Open a New Ticket Tickets (0)

Unable to create a ticket Support period has already expired on 2019-11-02 06:41:05.

Contacts us now to receive 10% off on your support renewal!

**Open a New Ticket**

Please fill in the form below to open a new ticket.

Email: john.doe@cartmega.com  
Client: John Doe

**Help Topic**  
Product Support

**Product Support Form**

**Product \***  
What product is this concerning?  
snappyVPS

**Domain \***  
Enter the URL where this product is being used with regards to this support request.

## LICENSE VERIFICATION

Verifies that your customer holds a valid license and so allow the provision of customer support. This makes it the ideal solution for sellers of software and other digital products on multiple markets online.

As seen on our own support ticket system!

Check out the video and online demo below:



**License Verification plugin for osTicket**  
(includes osTicket Motherload)

**VERIFY CLIENT'S LICENSE BEFORE PROVIDING SUPPORT**

**INSTALLATION** ✓  
**CONFIGURATION** ✓  
**TESTING** ✓

Visit [ostdemo.cartmega.com](http://ostdemo.cartmega.com) for a test drive.

[www.cartmega.com](http://www.cartmega.com)

**cart mega demo**

# SOLUTIONS TO PROBLEMS



**PLEASE NOTE:** You should also check the 'Common Problems' section in the readme file for possible errors and solutions, since it is more current.

**Problem:** The following message may appear in the Motherload plugin page:  
**Error: :**  
**ML: :motherload - Plugin syntax error: Plugin '... \plugin\_demo.php' failed syntax checking and was not loaded.**  
**Array**  
**([0] => Could not open input file:**  
**'.../plugins/demo/plugin\_demo.php')**

**Description:** As the message suggests, Motherload attempted to execute a syntax check on the plugin but could not open the file. This is not a permissions issue because the readability test was already checked before the syntax check. The problem here most probably lies with the web server being unable to execute the 'php -l <path-to-file.php>' command.

This usually happens on shared hosting or other installations where PHP suffers CLI execution limitations.

You can confirm the problem programmatically by executing `<php -l 'path-to-script.php'>` which should return 'Could not open input file: path-to-script.php'.

*Solution:* Since version 1.4.1, the user can disable syntax checking from the plugin page. When disabled, Motherload will not conduct syntax checking on your plugin scripts before trying to include and then execute them. This should not be a major setback since most errors are also caught and reported in the debugger when Motherload later attempts to include the script.

*Problem:* 'HTTP ERROR 500' in the browser, with the following error showing in the Motherload plugin page and/or in your web server log:  
**PHP Parse error: syntax error, unexpected 'const' (T\_CONST), expecting variable (T\_VARIABLE) in <PATH>/include/plugins/motherload/motherload.php on line 77**

*Description:* Class constant visibility modifiers were introduced in PHP 7.1.0. Any version before it will throw syntax error.  
PHP implementations may come with limitations in functionality even if the version is the same.

*Solution:* Check and upgrade the PHP version on your server.  
or  
Replace 'public const' into 'const' on the line mentioned in the error.

**Problem:** The following messages may appear in the Motherload plugin page:  
**Error: :**

**ML: :motherload - Plugin syntax error: Plugin '...\plugin\_demo.php' failed syntax checking and was not loaded.**

**Array**

```
(  
    [0] => 'php' is not recognized as an internal or external  
command,  
    [1] => operable program or batch file.  
)
```

**Or**

**Error: :**

**ML: :motherload - Plugin syntax error: Plugin '...\plugin\_demo.php' failed syntax checking and was not loaded.**

**Array**

```
(  
)
```

**Description:** As the message suggests, PHP is required but not being recognized. Sometimes the webserver, IIS in this example, is not correctly configured to run PHP.

**Solution:** Make sure PHP is working properly on your server.

Some IIS users have reported that having installed "Microsoft Visual C++ 2015 Redistributable Update 3", fixed their problem.

Their previous version of MSVC++ was making PHP itself fail (even typing '**php**' in cmd gave them an error).

---

## Why choose Cartmega?



### Affordable Pricing:

We maintain transparency in our pricing and we always remain competitive, so that you can reap the benefits.



### Security and Performance:

When developing our software, system security is our top priority, while never compromising on performance. Ordering from us is 100% safe and secure so you can rest easy. Your personal details are never shared, sold or rented to anyone either.



### 100% Satisfaction:

We insist that you love everything you buy from us. If you're unhappy for any reason whatsoever, just let us know and we'll bend over backwards to make things right again.



### World-Class Service:

All our products come with amazing service. Our online ticketing system and helpful staff will make sure of it.



### Money-Back Guarantee:

You get a full 30 days to get your money back, for all downloadable products. If it simply will not work on your setup and we cannot fix the problem then we'll cheerfully refund you every cent. For everything else, you get a full 14 day no-questions asked, money back guarantee.



### Easy Returns:

Returns are easy, simply log into your account and fill in the returns form for fast processing. We'll get you a refund in a snap!

*Let's not forget*

## The Essentials



### USER GUIDE

Full instructions are included with every product, which consist of installation and usage guidelines and other relevant information.



### LIFETIME UPDATES

With free updates for the lifetime of the product you need not worry about software maintenance. We got you covered!



### PREMIUM SUPPORT

All customers get access to world-class support via our online ticketing system for amazing after-sales service.

---



CONTACT US

## Still Can't Decide?!

Just in case you still have questions or not sure that what you've chosen is suitable, no worries. Contact us, we are here to help.

[Get In Touch](#)

### Software disclaimer

Software developed by cartmega.com is provided 'as is' without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of fitness for a purpose, or the warranty of non-infringement.

Without limiting the foregoing, cartmega.com makes no warranty that:

- the software will meet your requirements;
- the software will be uninterrupted, timely, secure or error-free;
- the results that may be obtained from the use of the software will be effective, accurate or reliable;
- the quality of the software will meet your expectations;
- any errors in the software obtained from the cartmega.com web site will be corrected.

Software and its documentation:

- could include technical or other mistakes, inaccuracies or typographical errors. cartmega.com may make changes to the software or documentation made available on its web site;
- may be out of date, and cartmega.com makes no commitment to update such materials;
- cartmega.com assumes no responsibility for errors or omissions in the software or documentation available.

In no event shall cartmega.com be liable to you or any third parties for any special, punitive, incidental, indirect or consequential damages of any kind, or any damages whatsoever, including, without limitation, those resulting from loss of use, data or profits, whether or not cartmega.com has been advised of the possibility of such damages, and on any theory of liability, arising out of or in connection with the use of this software.

The use of the software is at your own discretion and risk and with agreement that you will be solely responsible for any damage to your computer system or loss of data that results from such activities. No advice or information, whether oral or written, obtained by you from cartmega.com or from the cartmega.com web site shall create any warranty for the software.



© 2019 CartMega. All rights reserved.



[www.cartmega.com](http://www.cartmega.com)