

---

# **Argux Server Documentation**

***Release 0.0.1***

**Stephan Arts**

November 26, 2016



<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Getting Started</b>	<b>5</b>
2.1	Generate a config file . . . . .	5
2.2	Initialize database . . . . .	6
<b>3</b>	<b>Administration</b>	<b>7</b>
<b>4</b>	<b>Monitors</b>	<b>9</b>
4.1	ICMP Monitor . . . . .	9
4.2	DNS Monitor . . . . .	10
4.3	SNMP Monitor . . . . .	13
4.4	JMX Monitor . . . . .	14
<b>5</b>	<b>REST API</b>	<b>15</b>
5.1	Authentication . . . . .	15
5.2	Hosts . . . . .	15
<b>6</b>	<b>Command-Line Interface</b>	<b>17</b>
<b>7</b>	<b>Development</b>	<b>19</b>
7.1	ArguxServer.monitors . . . . .	19
7.2	ArguxServer.views . . . . .	19
<b>8</b>	<b>Indices and tables</b>	<b>21</b>



Contents:



---

# Introduction

---

Argux is a monitoring framework that is designed to work slightly different from most monitoring applications that you'll find on the web. It is designed with the following aspects in mind.

---

**Note:** Argux is still under heavy development, some of the statements below might still be a bit 'optimistic'. (eg. they will be true once there is a release) - These remarks also count as reminders to what it is that is being developed.

---

**Decentralisation** In short, most of the time you are not interested in ping times from your monitoring host to server X. You are interested in ping times from the offices in Paris and Madrid to the server in Berlin.

**REST API** It is impossible to prepare for any scenario of hosts or services that require monitoring. Therefor it is important that the interface to Argux is clear and easy to use. It will have client API's in several programming languages, but if all else fails you could control it with `CURL`.

The REST API is the center of the application, it is meant for use by external scripts but it is also used internally and as backend to the Web-interface.

**Encryption** Since Argux is using basic HTTP requests, you can simply configure your favourite reverse-proxy or SSL terminator to run all traffic via HTTPS instead of plain HTTP.

**Authorisation** All requests to Argux require authentication. A user is allowed to write metrics of one host, or many. Or is only allowed to read information.

**Example 1:** Basically, I only trust the user 'postgres' on the database-server to store any postgres related information.

**Example 2:** An agent running on server X is not able to save metrics about server Y, unless is is explicitly allowed.





---

## Getting Started

---

### 2.1 Generate a config file

Use *argux-server\_genconfig* to generate a configuration-file. This command asks you a number of questions about the preferred configuration before generating the configuration-file:

```
$ argux-server_genconfig

#####
###                               ###
### Argux Server configuration wizard.  ###
###                               ###
#####
```

Specify the location where the configuration-file should be created:

```
Config file location[./argux-server.ini]:
```

If the file already exists, you are asked if it okay to overwrite. Answering ‘y’ will continue the wizard, ‘n’ will terminate it:

```
File 'filename' exists, overwrite? ['y','n'] (Default: y):
```

If you do not plan to use Argux-Server over HTTPS, you can disable secure-cookies, it is advised to use secure-cookies:

```
Use secure cookies? (Enforce HTTPS) ['y', 'n'] (Default: y):
```

---

**Note:** If secure-cookies is enabled and you don’t use HTTPS, the system won’t work.

---

#### 2.1.1 Debugging

Do you want to enable debugging? (useful for development or troubleshooting):

```
Enable debugging? ['y', 'n'] (Default: n):
```

#### 2.1.2 WSGI Server

Pick the wsgi server:

```
WSGI Server? ['pserve','uwsgi'] (Default: pserve):
```

If you've picked pserve, you start argux-server like this:

```
pserve argux-server.ini
```

If you've picked uwsgi, it will start as followed:

```
uwsgi --ini-paste argux-server.ini
```

### 2.1.3 Database Configuration

Choose the database engine:

```
Choose Database Engine (mysql, pgsql, sqlite):
```

---

**Note:** Only sqlite and pgsql are implemented in this wizard at the moment. Other engines should work (like mysql), but you'd have to modify the configuration-file manually to do so.

---

#### Sqlite3

Select the sqlite3 db-path:

```
'database path: (/var/lib/argux-server/argux.sqlite)'
```

#### Postgresql

Enter database server:

```
'database server: (localhost)'
```

Provide a database name:

```
'database name: (argux)'
```

Provide a database user:

```
'database user:'
```

Enter the password:

```
'database password:'
```

## 2.2 Initialize database

Argux-Server supports a number of database backends:

```
$ argux-server_initdb ./argux-server.ini
```

---

## Administration

---

TODO



---

## Monitors

---

Monitors are executed by backend-processes to perform monitoring tasks. These backend-threads use the same REST API as everything else inside Argux, like registering metric-items and publishing values.

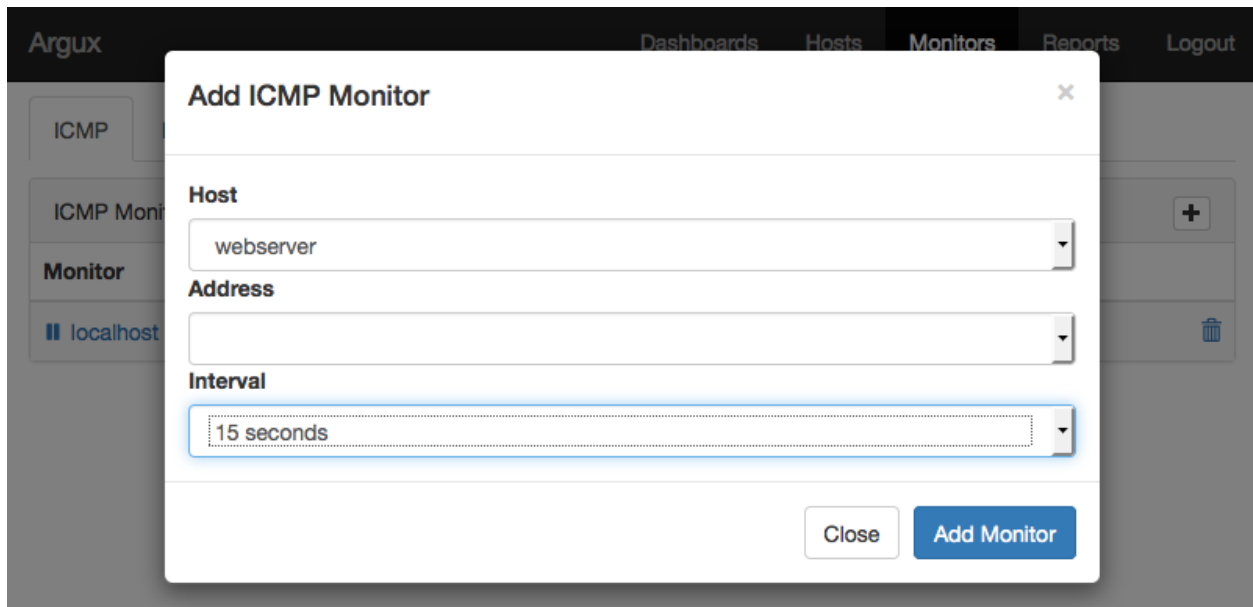
### 4.1 ICMP Monitor

The ICMP Monitor executes the *ping* command and performs an ICMP echo request to a specified host-address. The results of this command are stored in items associated with this host.

The screenshot shows the Argux web interface. At the top is a dark navigation bar with the 'Argux' logo on the left and links for 'Dashboards', 'Hosts', 'Monitors' (which is active), 'Reports', and 'Logout' on the right. Below this is a sub-navigation bar with tabs for 'ICMP', 'DNS', 'SNMP', and 'JMX'. The 'ICMP' tab is selected. The main content area displays 'ICMP Monitors' with a '+ ' icon in the top right corner. Below this is a table with two columns: 'Monitor' and 'Options'. The table contains one entry: 'localhost (127.0.0.1)' under the 'Monitor' column and 'interval: 15s' under the 'Options' column. A trash icon is visible at the end of the row.

Monitor	Options
localhost (127.0.0.1)	interval: 15s

### 4.1.1 Create a new ICMP Monitor



The screenshot shows the 'Add ICMP Monitor' dialog box in the Argux application. The dialog has a title bar with a close button (X). It contains three input fields: 'Host' with the value 'webserver', 'Address' which is empty, and 'Interval' with the value '15 seconds'. At the bottom right, there are two buttons: 'Close' and 'Add Monitor'.

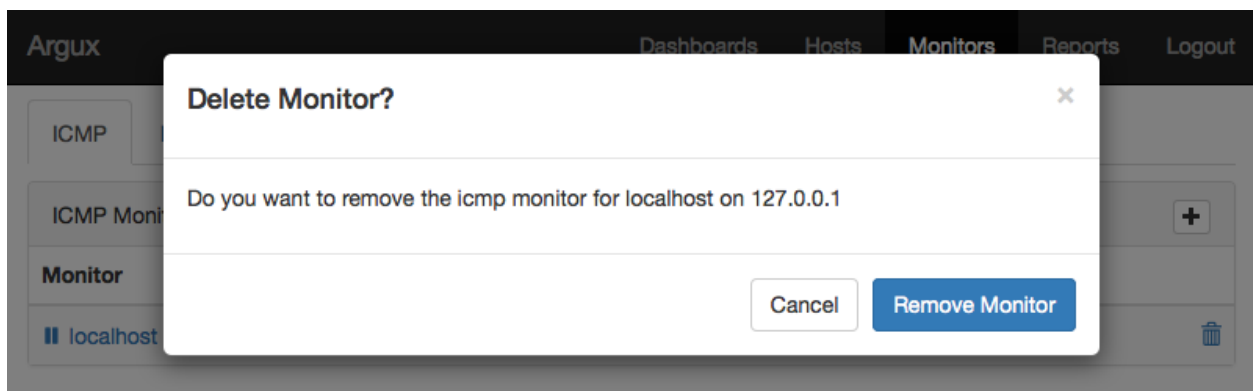
### 4.1.2 Delete an ICMP Monitor

A monitor can be deleted by clicking on the `trashcan` icon on the right.

---

**Note:** Consider pausing a monitor instead by clicking on the `pause` button on the right.

---



The screenshot shows the 'Delete Monitor?' dialog box in the Argux application. The dialog has a title bar with a close button (X). It contains a message: 'Do you want to remove the icmp monitor for localhost on 127.0.0.1'. At the bottom right, there are two buttons: 'Cancel' and 'Remove Monitor'.

## 4.2 DNS Monitor

The DNS Monitor can be used to monitor a DNS server and guard it's responses. It executes the *dig* command to retrieve specific information about a DNS record and stores those results with associated items.

ICMP

DNS

SNMP

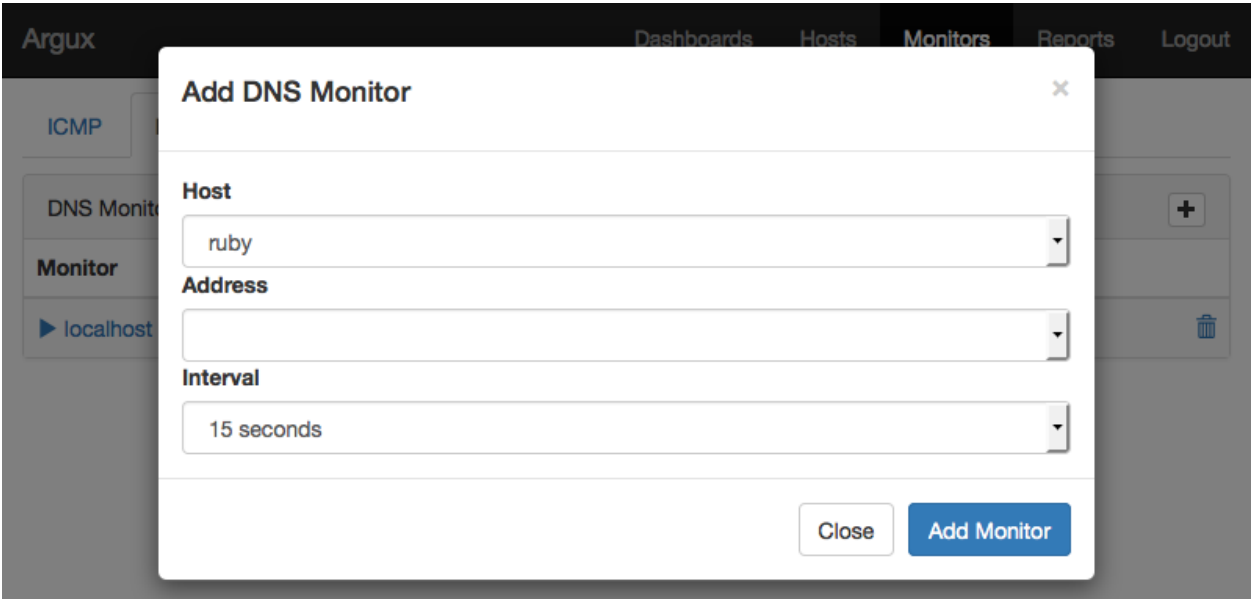
JMX

DNS Monitors		+
Monitor	Options	
▶ localhost (127.0.0.1)	interval: 15s	🗑

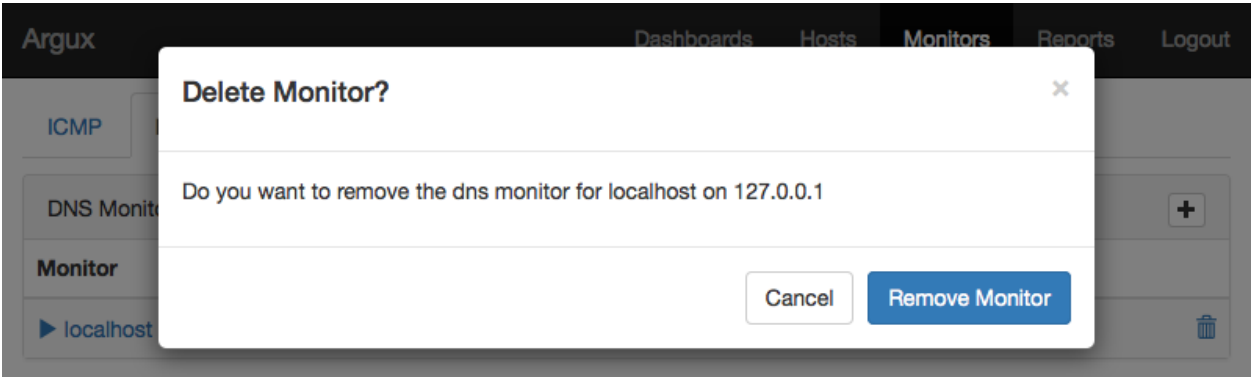




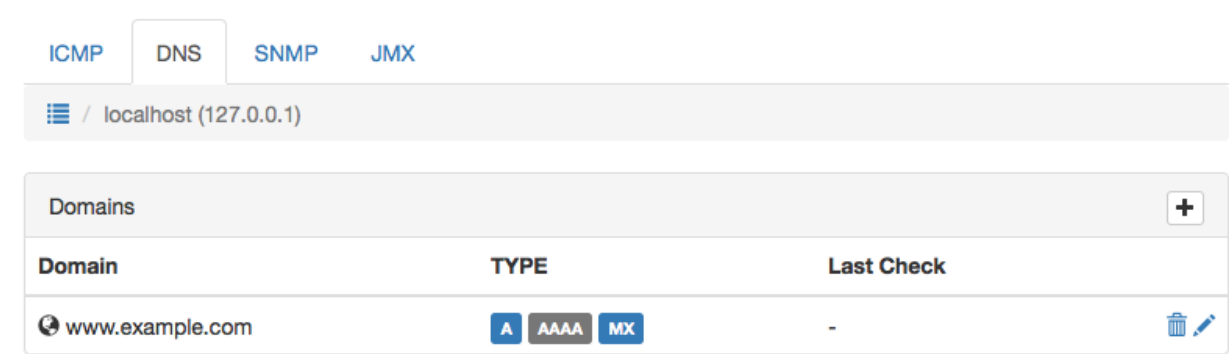
4.2.1 Create a new DNS Monitor



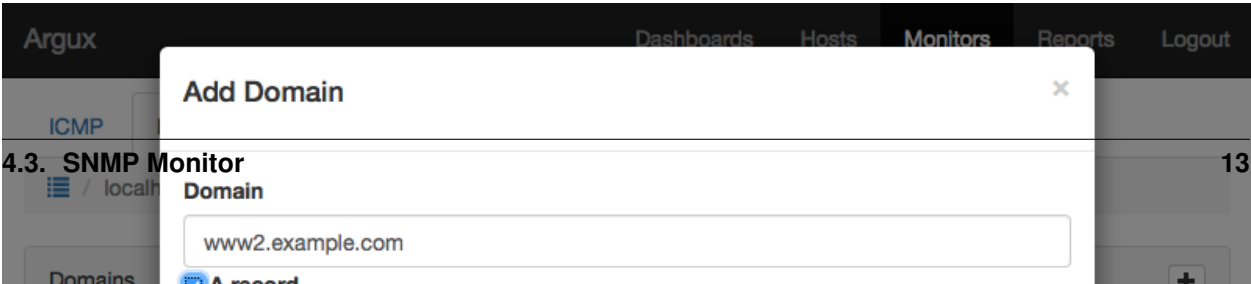
4.2.2 Delete an DNS Monitor



4.2.3 Check Monitored DNS domains



4.2.4 Add Monitored DNS domain



4.3. SNMP Monitor

---

**Note:** The SNMP Monitor is not yet implemented, any suggestions are welcome.

---

## 4.4 JMX Monitor

---

**Note:** The JMX Monitor is not yet implemented, it requires a java proxy that translates to the binary JMX protocol.

---

---

## REST API

---

The REST API is at the base of everything inside Argux. It is exposed so anyone can write/adapt their own scripts to integrate their metrics.

### 5.1 Authentication

First of all you need an authentication-token before you can do anything (ofcourse).

#### 5.1.1 Get Authentication-Token

##### CURL

The following script is an example using CURL:

```
#!/bin/bash
HEADER_FILE=`mktemp`
COOKIE_FILE=`mktemp`

curl -X POST \
  -c $COOKIE_FILE \
  -D $HEADER_FILE \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -d "{\"username\":\"jdoe\",\"password\":\"password\"}" \
  http://localhost/rest/1.0/login

CSRF_TOKEN=`cat $HEADER_FILE | \
  grep -i X-CSRF-TOKEN | \
  awk -F : '{ print $2 }'`
```

The Session Cookie and Cross-Site-Request-Forgery token must be preserved for any following requests.

### 5.2 Hosts

Section describing Host API.

### 5.2.1 CREATE Host

Create a host

#### CURL

Create a host using the following command:

```
curl -X POST \  
  -b $COOKIE_FILE \  
  -H "Content-Type: application/json" \  
  -H "X-CSRF-Token: $CSRF_TOKEN" \  
  -d "{  
    \"description\": \"Argux DEMO System\"  
  }" \  
  http://localhost/rest/1.0/host/HOSTNAME
```

### 5.2.2 READ Host

todo...

### 5.2.3 UPDATE Host

todo...

### 5.2.4 DELETE Host

Delete a host

#### CURL

Delete a host using the following command:

```
curl -X DELETE \  
  -b $COOKIE_FILE \  
  -H "Content-Type: application/json" \  
  -H "X-CSRF-Token: $CSRF_TOKEN" \  
  http://localhost/rest/1.0/host/HOSTNAME
```

---

## Command-Line Interface

---

...



---

**Development**

---

**7.1 ArguxServer.monitors**

**7.2 ArguxServer.views**





---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`