Aluminum

Nov 28, 2021

General Documentation

CHAPTER 1

What is Aluminum?

Aluminum is a lightweight, simple-to-install, and easy-to-use web server written entirely in Node.js, "an asynchronous event-driven JavaScript runtime ... designed to build scalable network applications." It includes not only a traditional static/dynamic web server, but also various other features that simplify the web development process, such as authentication and network-based cryptography platforms.

CHAPTER 2

Setup and General Knowledge

Note: In this documentation, `/` refers to the root directory of the Aluminum repository, except as otherwise noted.

To install Aluminum, simply clone the git repository, install the dependencies automatically with `npm`, and copy the default user files from the `/defaults` directory to `/usr`. Starting the server is as simple as typing `npm start` into your terminal.

See also:

Please see the installation guide for more detailed instructions on installing Aluminum.

Aluminum is controlled by JSON configuration files. These files are found in the `/usr/prefs` directory.

2.1 The listen.json file

The `listen.json` file, located at `/usr/prefs/listen.json` determines which listening addresses Aluminum uses for different features. The file contains a single `listenAddresses` object, described below.

A default configuration of `listen.json` is located at `/defaults/prefs/listen.json`.

`listenAddresses`

Object Properties

- **`wire`** (*`wirePorts`*) – Listening address for Aluminum Wire

`wirePorts`

Object Properties

- **`HTTP`** (*`String`*) – Port to use for Aluminum Wire over HTTP
- **`HTTPS`** (*`String`*) – Port to use for Aluminum Wire over HTTPS

Warning: Wire does not yet support HTTPS.

CHAPTER 3

Features

Aluminum offers several unique and useful features:

- **Highly Customizable:** Use an official extension, a community-contributed one, or make your own.
- **Integrated Authentication Server:** Verify end users' identity without leaving the Aluminum platform.
- **PHP support:** If you're uncomfortable using Node.js for server-side scripting, Aluminum is also compatible with PHP.
- **And more:** Remote system resource monitor, network based time synchronization, math rendering, simplified cryptography, etc.

3.1 Aluminum Wire

Aluminum Wire is a highly-customizable and extensible static file server that also includes support for dynamic files (using both Node.js and PHP).

Note: Dynamic file support is not yet available

3.1.1 Configuration

The configuration file for Aluminum Wire is read from `/usr/prefs/wire.json`. The supported configuration options are described below.

`wireConfig`

Object Properties

- **`protocol`** (*`String`*) – The protocol for Wire to use, either `http` or `https`

Note: `protocol` is not yet implemented.

- **`indexRedirect`** (*`Boolean`*) – Whether to respond to requests for a directory by serving the `index.html` file in that directory (if it exists; otherwise, a 404 response code will be served)
- **`errorPages`** (`wireErrPgs`) – Configuration options for the error pages served by Wire

`wireErrPages`

Object Properties

- **`notFound`** (*`errPageConf`*) – Describes the page that Wire should serve if a resource cannot be found
- **`serverError`** (*`errPageConf`*) – Describes the page that Wire should serve in the event of an internal server error

`errPageConf`

Object Properties

- **`URI`** (*`String`*) – The URI of the file to serve, given relative to `/src/wire/main.js`
- **`encoding`** (*`String`*) – The encoding of the file to serve. If the MIME type that corresponds to `URI` is not `text`, this property will be ignored.

3.1.2 Serving Files

Wire serves files from `/usr/resources/wire/serve/`.

Client Cache Support

When Wire serves a static file, it retrieves the date that the file was last modified and sends this information in the `Last-Modified` HTTP header.

Wire automatically reads the `If-Modified-Since` HTTP header from the request when serving static files. If the requested resource exists and has not been modified since the time indicated by `If-Modified-Since`, a `304 Not Modified` response code is served, indicating that the cached version of the file is up-to-date. Otherwise (or if the request does not contain an `If-Modified-Since` header), the resource is served.

Error Handling

The following table shows the types of errors that Wire will handle when serving files:

Problematic Function	Error Code	HTTP Response Code	Response Body
`fs.readFile`	Any	404	The file at `wireConfig.errorPages.notFound.URI`
`fs.stat`	Any	500	The file at `wireConfig.errorPages.serverError.URI`

Caution: Errors with reading the Wire configuration file (e.g., the file cannot be accessed at `/usr/prefs/wire.json`, a required configuration option is missing) or error pages (e.g., the error page cannot be accessed) will raise an exception and cause Wire to crash.

Error Page Variables

In error pages with MIME type `text`, information about the server and error may be included in the response sent. Variables may be inserted anywhere within the file and are surrounded by the dollar (`$`) symbol.

Note: If two variables need to be inserted in an error page back-to-back, then each variable should have its own set of `$` symbols.

Note: Variables are case-sensitive.

The possible variables that may be used within an error page are described below. Each variable may be used zero, one, or multiple times.

Variable	Description
`$requrl$`	The request URL
`$adjrequrl$`	The request URL with "index.html" appended, if `wireConfig.indexRedirect` is set to `true`. Otherwise, this is the same as `$requrl$`. This variable will only function in the event of an `fs.readFile` error.
`$osplatform$`	The operating system platform of the server. See os.platform().
`$ostype$`	The operating system type of the server. See os.type().
`$osversion$`	The operating system release of the server. See os.release().
`$port$`	The port on which the Wire server is listening.
`$errcode$`	The code of the error. See error.code.
`$errno$`	The number of the error. See error.errno.
`$errmessage$`	The error message. Note that this may contain information such as the absolute path to a resource on a server. See error.message.

Index

J