Zendbit Documentation Documentation

Release latest

Aug 03, 2020

Contents

1 Cargo Server **3**
- 1.1 Supported platforms . . . 3
- 1.2 Installation . . . 3
- 1.3 Global Configuration . . . 4
- 1.4 User Creation . . . 4

2 Cargo Client **5**
- 2.1 Supported platforms . . . 5
- 2.2 Installation . . . 5
- 2.3 Running Modes . . . 6

CHAPTER 1

Cargo Server

1.1 Supported platforms

It is currently supported and maintained for the following platforms and architectures

Operative System	Architecture
Linux 2.6.23 or later with glibc	amd64, 386, arm, arm64

1.2 Installation

1.2.1 On Premise

The following steps are required to install the On Premise version:

1. Environment creation and configuration
2. Cargo Server Installation
3. Apache or Nginx web server installation and configuration
4. Enabling the corresponding ports in the Firewall

These tasks must be performed by the solution implementer and are included with the purchase of the on-premise license.

1.2.2 Cloud

The Cloud version is installed by deploying the Cargo Server image.

1.3 Global Configuration

Cargo Server has global configuration parameters and client-specific parameters. The global configuration parameters allow you to set limits on the number of simultaneous connections and the total bandwidth used. These configuration parameters prevail more than the client-specific parameters.

1.3.1 Global Parameters

- **Global Upload Bandwidth**: Total upload bandwidth assigned to the server
- **Global Download Bandwidth**: Total download bandwidth assigned to the server.
- **Private IP**: Private IP of the interface to be used
- **Public IP**: Public IP of the link to be used.
- **Initial Port**: Port on which the first connection will be established. The range of ports used will be given by the Initial Port + Total connections count.
- **Total connections**: Global number of simultaneous connections enabled.

1.4 User Creation

Within the User ABM section, it is possible to manage the credentials for new clients and specify the connection parameters for each one, allowing greater granularity in the use of the available bandwidth.

1.4.1 Client Parameters

- **Username**: Client's user name.
- **Password**: Client's password.
- **Path**: Absolute path where the client's root directory will be found.
- **Upload Speed**: Default upload speed of the transfers.
- **Download Speed**: Default download speed of the transfers.
- **Upload Bandwidth**: Total upload bandwidth assigned to the client.
- **Download Bandwidth**: Total download bandwidth assigned to the client.
- **Enable**: Enable or disable a client

CHAPTER 2

Cargo Client

2.1 Supported platforms

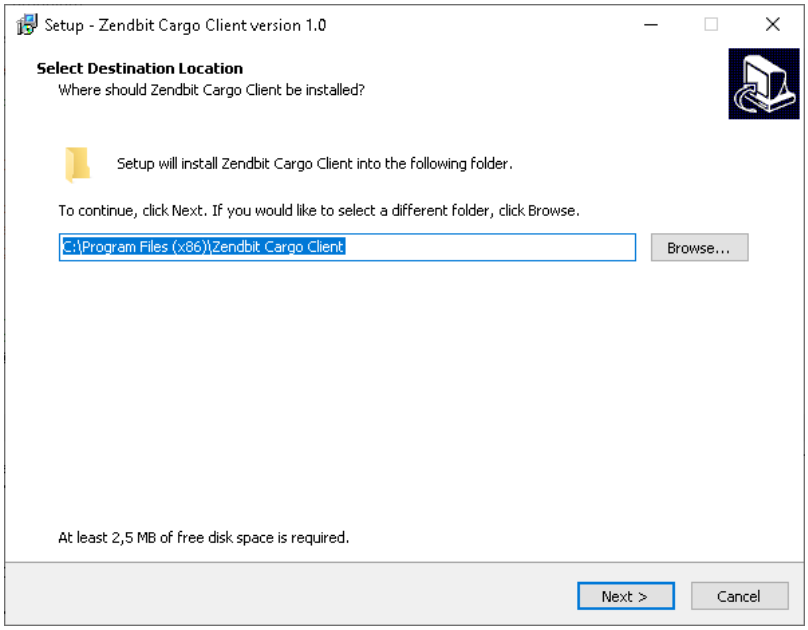
It is currently supported and maintained for the following platforms and architectures

Opertive System	Architecture
Linux 2.6.23 or later with glibc	amd64, 386, arm, arm64
macOS 10.11 or later	amd64
Windows 7, Server 2008R2 or later	amd64, 386

2.2 Installation

2.2.1 Windows

Cargo Client can be installed by downloading the installer and following the running steps.



2.2.2 GNU/Linux

```
# cd /opt/
# wget https://download.zendbit.com/cargoclient-v1.linux.tar.gz
# tar -vxzf cargoclient-v1.linux.tar.gz
```

2.2.3 OSx

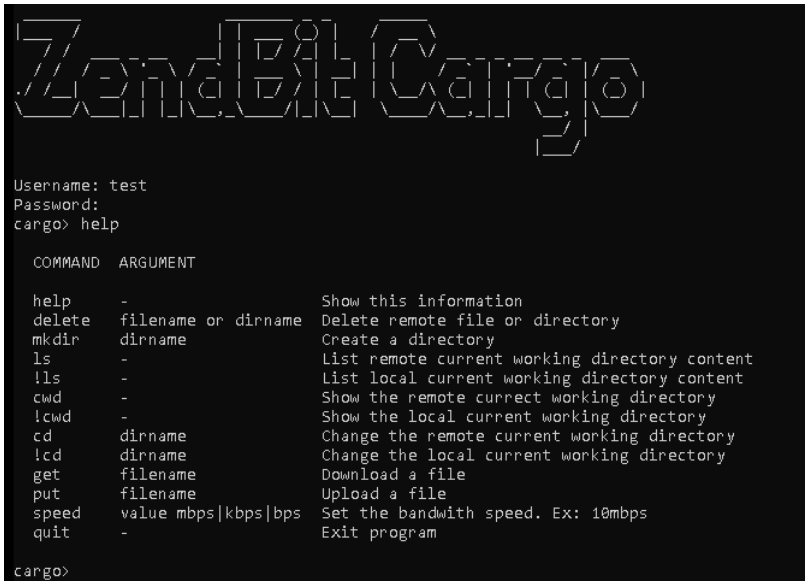
```
# cd /opt/
# wget https://download.zendbit.com/cargoclient-v1.osx.tar.gz
# tar -vxzf cargoclient-v1.osx.tar.gz
```

2.3 Running Modes

2.3.1 Interactive Client

The interactive client is very similar to a command line FTP client. The client mode is run as follows:

```
# cargoclient --mode=client --host=<url>
```



2.3.2 Automatic Synchronization

The "Automatic Synchronization" run mode allows to leave the Cargo Client running and synchronize a remote folder with a local one. The synchronization mode can either upload files to a remote location or discover new files on a remote server for downloading.

Running

```
# cargoclient --mode=datasync --config=nombre_archivo
```

Configuration File

The configuration file contains the necessary information to synchronize the folders

```
{
    "sync_name": "Name of the Job",
    "sync_mode": "download",
    "start_time": "10:00pm",
    "end_time": "12:00am",
    "local_dir": "/CagoPlay/test/",
    "remote_dir": "/",
    "username": "test",
    "password": "test",
    "speed": "10mbps",
    "hostname": "https://cargo.cexar.io",
    "sleep": 3,
    "events": {},
}
```

- **sync_name**: Free text representing the name of the Job
- **sync_mode**: "download" o "upload"
- **start_time** (Optional): Time when the job will start sending/receiving files

- **end_time** (Optional): Time when the job will finish sending/receiving files
- **speed**: Speed you will try to negotiate to send/receive files. It can be expressed in mbps, kbps or bps
- **sleep**: Timeout for detecting new files
- **events** (Optional): Callback configuration at start, update and end or error of a transfer.

Events

Events are useful for integration with other systems. Each time an event is triggered, the http action specified in the configuration is performed.

- **onStart**: This event is triggered at the start of a file transfer. Variables: FILE_NAME, TRANSFER_ID, SYNC_NAME, SYNC_MODE
- **onFinish**: This event is triggered after the successful completion of the transfer. Variables: FILE_NAME, TRANSFER_ID, SYNC_NAME, SYNC_MODE, AVG_SPEED, DURATION, LOST_FRAMES, AVG_RTT
- **onError**: This event is triggered after an unsuccessful transfer. Variables: FILE_NAME, TRANSFER_ID, SYNC_NAME, SYNC_MODE, ERROR
- **onUpdate**: This event is triggered 1 time per second during the entire transfer. Variables: FILE_NAME, TRANSFER_ID, SYNC_NAME, SYNC_MODE, PROGRESS, SPEED

Adding Callbacks for Events in the Configuration

- **Supported http methods**: POST, GET. The POST method requires the body configuration parameter.
- **Variables**: Each event has its own variables, which can be used to set up the url as well as the body of the request

```
"events": {
    "onStart": {
       "method": "POST",
       "url": "https://api.my-system.com/cargo",
       "body": { "foo": "{{FILE_NAME}}" }
    },
    "onError": {
       "method": "GET",
       "url": "https://api.my-system.com/cargo/{{FILE_NAME}}",
    }
 }
```