# Summary

This document describes the REST Web Service interface exposed by the directory server.

# Requests destination

All HTTP requests will be sent to:

[http://sso.zeepro.com/*function?parameters*](http://sso.zeepro.com/function?parameters)

Except for some mentioned exceptions, requests will be sent by POST HTTP method. The same requests sent by GET will return an HTML from, allowing the call of the function with a HTML browser.

In production, traffic will be tunneled through SSL for information can be exchanged securely.

# App interface

Successful requests will return a 200 HTTP error with a possible answer in the body, depending on the function.

Unsuccessful requests will return a custom HTTP error code specified for each function.

## Change password

Change an existing account password.

Syntax: /changepassword.ashx

The changepassword function will be called by POST method with the following URL-encoded parameters:

* *email*
* *old\_password*
* *new\_password*

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed

## Confirm account

Account creation confirmation

Syntax: /confirmaccount.ashx

The confirmaccount function will be called by POST method with the following URL-encoded parameters:

* *email*
* *code*

Note:

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 435: Too many requests
* 437: Account already exist (if already confirmed)

## Connect

Log user connection to a printer and set calibration warning message state.

Syntax: /connect.ashx

Return value:

none

Note:

The connect function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *printersn* : printer serial number
* *calibrationwarningmessage*: <"yes"|"no"> (optional)

HTTP error code:

* 432: Missing parameter
* 442: Unauthorized user

## Create account

Create a new account.

Syntax: /createaccount.ashx

The createaccount function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *language (optional)*
* *optin (optional)*

Note:

Requests will be limited to 5, every 5 seconds, by IP. At creation, an email with a confirmation code (4 random numbers) will be send to the email address using Mandrill. The email template (subject + html template) will be stored in the database. An account must be confirmed (see below) to be used; login with an unconfirmed account will return a “434: Login failed” error.

The language identifier will follow the ISO 639. English (“en”) will be used if it’s not specified.

If the user accepts to receive commercial e-mail solicitations, the “optin” variable value should be set to “on” (any other value or no value will be interpreted as “off”).

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 435: Too many requests
* 437: Account already exist (only if confirmed)

## Delete account

Delete an existing account.

Syntax: /deleteaccount.ashx

The deleteaccount function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed

## Get local IP

Get the local IP of a given printer.

Syntax: /getlocalip.ashx

Return value:

Json formated string (RFC 4627) containing an array of active printers associated with the account:

{state: "*state*"*,* localIP: "*localIPaddress*"}

*state*: ok | unknown

Note:

The getlocalip function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter

## List printers

List active printers associated with an account.

Syntax: /listprinter.ashx

Return value:

Json formated string (RFC 4627) containing an array of active printers associated with the account:

[{printername: *printername1,* localIP: *localIPaddress1,* URL: *URL1,* token: *token1,* currentversion: *current software version,* nextversion: *next software version*, lastaccess: *last access date*, calibrationwarningmessage:<"yes"|"no">},

{printername: *printername2,* localIP: *localIPaddress2,* URL: *URL2,* token: *token2,* currentversion: *current software version,* nextversion: *next software version*, lastaccess: *last access date*, calibrationwarningmessage:<"yes"|"no">},

…]

Note:

The login function will be called by POST method with the following URL-encoded parameters:

or

* *email*
* *password*

or

* *token\_id*: token given by /login

Listed printers will be taken from the “active” directory (see register function).

If current or next software version are unknown, they won’t be provided.

Lastaccess return the ISO-8601 formatted date of the last connection to the printer (see /connect.ashx). In conjunction with calibrationwarningmessage, it allows the display of the warning message about calibration shown in a pop-up when the user connects to the printer. If last access date is unknown, it won’t be provided.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed

## Login

Check login credentials.

Syntax: /login.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *optin (optional)*

The optin status of the account is updated accordingly to the optin parameter if it is specified.

Return value:

Returns HTTP code 202: Accepted, if the credentials are valid and a Json formatted string - RFC 4627 – containing a 40 characters long token with a 24 hours validity:

{“token\_id”: *token\_id*}

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 439: Unknown email

## Optin

Get or set account optin status.

Syntax: /optin.ashx

The optin function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *optin (optional)*

Return value:

A Json formated string (RFC 4627) containing the optin status of the account:

{"optin":”on”}

or

{"optin":”off”}

Note:

Requests will be limited to 5, every 5 seconds, by IP.

If the optin parameter is present, the optin status of the account will be set to the given value (“on” of “off” - any other value or no value will be interpreted as “off”).

In all cases, the function will return the optin status (after being set if the optin parameter is present).

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 435: Too many requests

## Send email

Allow the printer to send application level email to the user.

Syntax: /sendmail.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *address (coma separated cc mail address)*
* *subject*
* *htmlbody*

Emails will be sent using Mandrill and will be limited to 3 messages every 3 minutes by email address.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 435: Too many requests

## Send password

Reset forgotten password to the associated email address.

Syntax: /sendpassword.ashx?email=*email*

* *email*

Note:

The function will be called by GET method with URL-encoded parameter.

Sending will be limited to 3, every 10 minutes by email address.

The function will reset the password by creating a random one (8 random chars and numbers) and sending it by mail using Mandrill. The email template (subject + html template) will be stored in the database.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 435: Too many requests

## Test account

Test if an account already exists.

Syntax: /testaccount.ashx

The testaccount function will be called by POST method with the following URL-encoded parameters:

* *email*

Return value:

Json formated string (RFC 4627):

{"account":”exist”}

if already exists

{"account":”unknow”}

otherwise

Note:

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 435: Too many requests

# Printer interface

## Add printer

Add a printer to an account list of association.

Syntax: /addprinter.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *printersn* : printer serial number (see Zim REST web service Printer type function)
* *printername* : printer user chosen friendly user name

If the printer is already registered it will be moved to the new account. The printer name will be stored / updated if specified.

If the account doesn't exist it will be created.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 436: Unknown printer

## Error logging

Send error messages to the SSO.

Syntax: /errorlog.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number
* *printertime*: *date/time - ISO-8601 formatted*
* *level*: 4-critical; 3-error; 2-warning; 1-info; 0-debug
* *code*: see “/API/Zim error code.docx”
* *message*: any further information that could help…

Error messages will be logged in a table containing also a date/time field generated by the SSO server and a message number (starting from 1 and incrementing by 1).

Each insertion will be preceded by a deletion keeping the table from storing more than 100.000 messages (the needed indexes will be properly created and maintained).

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter

## Get user info

Get current user personal information.

Syntax: /getuserinfo.ashx

Return value:

Json formatted string (RFC 4627):

{"country": *country*, "city": *city*, "birth\_date": *birth\_date*, "why": *why*, "what": *what*}

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login

The birth date will be ISO-8601 formatted.

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 442: Unauthorized user

## Grant user

Allow a user to access the printer.

Syntax: /grantuser.ashx

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *printersn*: printer serial number (its Ethernet MAC address without colon - see Zim REST web service Printer type function)
* *user\_email*: user mail
* *user\_name*: user name
* *message*: personal message sent to the granted user (optional)
* *account*: <"yes"|"no">
* *manage*: <"yes"|"no">
* *view*: <"yes"|"no">

If the account doesn't exist it will be created and a confirmation email will be sent.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer
* 442: Unauthorized user

## List rendezvous

At start-up (if its network configuration is defined) or after each network configuration change, all printer will register to the directory server using the following function:

Syntax: /listrendezvous.ashx

Return value:

List of the available rendezvous servers in order of preference (Json formatted string - RFC 4627):

[*subdomain1*,

*subdomain2*,

…]

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number (its Ethernet MAC address without colon - see Zim REST web service Printer type function)
* *currentversion*: current software version (optional)

The IP address of the printer server will be gathered from the request itself. A request to freegeoip.net (<https://freegeoip.net/>) will allow to fetch approximate geographical coordinates and to determine the nearest available rendezvous servers.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer

## List users

Retrieve users allowed to access the printer:

Syntax: /listuser.ashx

Return value:

List of the users allowed to access the printer (Json formatted string - RFC 4627):

[{email: *user\_email1*, name: *user\_name1*, account: <"yes"|"no">, manage: <"yes"|"no">, view: <"yes"|"no">},

{email: *user\_email2*, name: *user\_name2*, account: <"yes"|"no">, manage: <"yes"|"no">, view: <"yes"|"no">},

…]

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *printersn*: printer serial number (its Ethernet MAC address without colon - see Zim REST web service Printer type function)

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer
* 442: Unauthorized user

## New version

Notifies next software version.

Syntax: /newversion.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number
* *nextversion*: next software version (after reboot)

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter

## Redirect

Redirect the caller to the printer URL previously stored in the cache (Youtube parameter).

Syntax: /redirect.ashx?...state=*printersn*…

Note:

The function will be called by GET method with at least then URL-encoded parameters:

* state : the serial number of the printer

and will issue an 301 redirection to the printer URL stored in cache, with all parameters (including those of the redirection URL itself).

Ex : <http://sso.zeepro.com/redirect.ashx?state=F48000000&code=xxxxxxxx> will redirect to http://printerCachedUrl?state=F48000000&code=xxxxxxxx

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer
* 440: No redirection

## Redirect Facebook

Redirect the caller to the printer URL previously stored in the cache (Facebook parameter).

Syntax: /redirect.ashx?...sn*printersn*…

Note:

The function will be called by GET method with at least then URL-encoded parameters:

* sn: the serial number of the printer

and will issue an 301 redirection to the printer URL stored in cache, with all parameters (including those of the redirection URL itself).

Ex : <http://sso.zeepro.com/redirect.ashx?snF48000000&code=xxxxxxxx> will redirect to http://printerCachedUrl?sn=F48000000&code=xxxxxxxx

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer
* 440: No redirection

## Revoke user

Deny a user to access the printer.

Syntax: /revokeuser.ashx

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *printersn*: printer serial number (its Ethernet MAC address without colon - see Zim REST web service Printer type function)
* *user\_email*: user mail

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer
* 442: Unauthorized user

## Set user info

Set current user personal information.

Syntax: /setuserinfo.ashx

Return value:

None

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *country*: country
* *city*: city
* *birth\_date*: birth date (ISO-8601 formatted)
* *why*: why
* *what*: what

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 442: Unauthorized user

## Time

Allow the printer to adjust its internal clock.

Syntax: /time.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number

Return value:

*date/time of the server - ISO-8601 formatted*

Note:

Calls will be limited to 1 every minute by printer serial number.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 435: Too many requests

## URL cache

Adds an entry in the URL cache.

Syntax: /url.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number
* *url*: URL to return to with the OAuth 2.0 callback

This function will associate in a cache a printer’s serial number with an URL. This cache will be cleared every 20 minutes.

HTTP error code:

* 432: Missing parameter
* 436: Unknown printer

## User access

Allow the printer to determine the accesses granted to the current user.

Syntax: /useraccess.ashx

Return value:

Access level of the user (Json formatted string - RFC 4627):

{email: *user\_email*, name: *user\_name*, account: <"yes"|"no">, manage: <"yes"|"no">, view: <"yes"|"no">}

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token*: token given by /login
* *printersn*: printer serial number (its Ethernet MAC address without colon - see Zim REST web service Printer type function)

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 442: Unauthorized user

## User storage

The SSO will provide storage for user models. At printer level, a round-robin like cache mechanism based on the use of ETag will reduce the impact of remote storage.

Per user, each model is uniquely identified by its name. A model can be represented by:

* one or two 3D files
* an undefined number of prints, identified by a date, each with a G-code file and a time-lapse.

The storage interface manages no compression mechanism; the files will have to be compressed / decompressed by the printer.

### Add 3D file(s)

Store the 3D model associated file(s).

Syntax: /userlib/add3dfile

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *id*: model id
* *archive*: Zip compressed file including all files needed to be saved

Note:

The archive will contain the following files:

* model1.stl and, optionally, model2.stl: STL 3D files (max 100MB each)
* Or model.stl or model.amf: AMF 3D file (max 100MB)
* image.jpg: rendered image (max 1MB)
* metadata.json: text file containing a Json formatted string (RFC 4627) with the following information:
  + file1etag: first 3D file etag
  + file2etag: second 3D file etag (optional)

If the archive exceeds 512Kb it will have to be split in 512Kb chunks labeled as follow:

*archive\_name*.zip.*chunk\_number*.*number\_of\_chunks*

ex.:

* 3dfile.zip.1.3
* 3dfile.zip.2.3
* 3dfile.zip.3.3

The archive mustn’t exceed 80 chunks.

The function returns “partial” until all chunks have been received.

Chunks must be uploaded within 2 minutes delay.

Pre-existing files will be replaced.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 443: Unknown model
* 450: File too big

### Add print

Store a 3D model print associated G-code and time-lapse.

Syntax: /userlib/addprint

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *id*: model id
* *date*: print date (ISO 8601)
* *archive*: Zip compressed file including all files needed to be saved

Note:

The archive will contain the following files:

* *gcode*: G-code file (20MB limit)
* *time-lapse*: time-lapse video file (5MB limit)
* *img*: time-lapse video snapshot (1MB limit)
* *metadata.json*: text file containing a Json formatted string (RFC 4627) with the following information:
  + *description*: Json string including the description, the preset name and all information needed to reproduce the print
  + *gcodeetag*: G-code file etag
  + *time-lapseetag*: time-lapse video file etag

If the archive exceeds 512Kb it will have to be split in 512Kb chunks labeled as follow:

*archive\_name*.zip.*chunk\_number*.*number\_of\_chunks*

ex.:

* printfile.zip.1.3
* printfile.zip.2.3
* printfile.zip.3.3

The archive mustn’t exceed 80 chunks.

The function returns “partial” until all chunks have been received.

Chunks must be uploaded within 2 minutes delay.

Pre-existing files will be replaced.

HTTP error code:

* 432: Missing parameter
* 443: Incorrect parameter
* 450: File too big

### Create

Create a model.

Syntax: /userlib/create

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *name*: model name

Return value:

* *id*: model id

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 442: Unauthorized user
* 444: Model already exists

### Delete

Delete all information related to a model.

Syntax: /userlib/delete

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *id*: model id

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 442: Unauthorized user
* 443: Unknown model

### Delete print

Delete a print.

Syntax: /userlib/delete

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *id*: model id
* *date*: print date (ISO 8601)

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 442: Unauthorized user
* 443: Unknown model
* 445: Print unknown

### Get

For internal use; not to be documented.

Syntax: /userlib/get.ashx

### List

Get current user model list.

Syntax: /userlib/list.ashx

Return value:

Json formatted string (RFC 4627):

[{"id": *model id*,

"name": *model name*,

"3dfile1": *first 3D file URL |* "uploading" *(if exists)*,

"3dfile1etag": *first 3D file entity tag (if exists)*,

"3dfile2": *second 3D file URL |* "uploading" *(if exists)*,

"3dfile2etag": *second 3D file entity tag (if exists)*,

"print": [{"date": *ISO 8601 print date*,

"description": *Json string including the description, the preset name and all information needed to reproduce the print*,

"gcode": *G-code file URL |* "uploading",

"gcodeetag": *G-code file entity tag (if exists)*,

"imglink": *image file URL |* "uploading"*,*

"videolink": *time-lapse URL |* "uploading"*,*

}…] *(if exists)*

…]

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login

A model may have only a 3D

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 442: Unauthorized user

### Rename

Rename a model.

Syntax: /userlib/rename

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *token\_id*: token given by /login
* *id*: model id
* *name*: new model name

Return value:

* *id*: model id

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 442: Unauthorized user
* 443: Unknown model

# Rendezvous servers interface

## Rendezvous status

All Rendezvous servers will push their statistics every minute in order to inform the directory server about their availability.

Syntax: /setrendezvousstatus.ashx

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *bandwidth*: used bandwidth (in Mo/s)
* percentage: used bandwidth percentage (integer)

The IP address and the host name of the server will be gathered using the request itself. A request to freegeoip.net (<https://freegeoip.net/>) will allow to fetch approximate geographical coordinates.

Return value:

None

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter

## Set active printer

Each time a printer connects or refreshes its connection to a rendezvous server, that last will issue the /setactiveprinter function to the directory server.

Syntax: /setactiveprinter.ashx

Note:

The function will be called by POST method with the following URL-encoded parameters:

* *printersn* : printer serial number (see Zim REST web service Printer type function)
* *local*IPaddress : the IP address used by the printer within the its domestic network (e.g. : 192.168.0.18)
* *token*: security code emitted by the rendezvous server
* *port*: the port number dedicated to the printer by the Rendezvous server

The IP address and the host name of the server will be gathered form the request itself.

Registered printer will be add in an “active” directory. If a printer didn’t have refreshed its registration within the last 20 minutes its will be deleted from the “active” directory.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 436: Unknown printer

# Manufacturing interface

## Create printer

This function will allow a specific user to create a new printer.

Syntax: /createprinter.ashx

This function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *printers*: Json formatted string (RFC 4627) containing an array

[{manufactured:*date/time of the manufacturing - ISO-8601 formated*, type: *printer type name*, profile: URL du profile, hw\_version: identifiant de série, EAN: *EAN*, UPC: *UPC*, rangecode: *rangecode*, serialnumber: *MAC address*}, …]

Return value:

Json formatted string (RFC 4627) containing an array:

[{serialnumber: *MAC address,* result: *code*}, …]

Where *code* could be:

* “test” if rangecode value equal to “testreange” (for testing purposes)
* “unavailable range” if the rangecode is incorrect (syntactically or regarding the counter value)
* “incorrect MAC” if the MAC address is outside the range
* “already exist” if the printer address already exists
* “ok” if a new printer has been created

Note:

The email must be “zeeproserial@zeepro.com”.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed

## MAC addresses range request

This function will allow a specific user to request ranges of MAC addresses from the medium range allocated to Zeepro by IEEE (e.g. F4:0E:11:80:00:00 - F4:0E:11:8F:FF:FF).

Syntax: /requestmacrange.ashx

This function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *rangesize*: number of requested addresses

Return value:

The 12 hexadecimal digits of the first address, followed by the 12 hexadecimal digits of the last address, followed by the 6 first characters of the SHA1 hash of the 24 preceding followed by “ zeepro “ ; ex. :

* First available address: F40E118008F1 (the one just following the counter)
* Requested number of address: 100 (in decimal base)
* Last address: F40E11800954 (F40E118008F1 + 64 – 1 in hexadecimal base)
* String to be hashed: F40E118008F1F40E11800954zeepro
* SHA1 hash: 7833FB09352A76EFB1813C18ABA112EDAFF4AD93
* 6 first characters 7833FB
* Returned value: F40E118008F1F40E118009547833FB

Corresponding to the range F4:0E:11:80:08:F1-F4:0E:11:80:09:54.

Note:

The email must be “zeepromac@zeepro.com”.

The address range will be actually reserved and usable only when the call to reservemacrange.ashx function (see below) will be completed.

This function issues addresses based on a counter updated by the reservemacrange.ashx function (see below).

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 435: Too many requests

## MAC addresses range reservation

This function will allow a specific user to confirm the reservation of a previously requested ranges of MAC addresses.

Syntax: /reservemacrange.ashx

This function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *rangecode*: value returned by the /requestmacrange.ashx function

Return value:

A simple text/plain “ok” if everything is all right.

Note:

The purpose of this confirmation stage is to provide a better security level and to avoid allocation losses due to communication failures.

This function will:

* Verify the code correctness (if not a 438 error code will be return)
* Verify that the first address of the range is just above the counter
* Update the counter to the last address of the range

The email must be “zeepromac@zeepro.com”.

Requests will be limited to 5, every 5 seconds, by IP.

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 435: Too many requests
* 438: Unavailable range

# Service (slicing)

## Service server interface

### Service registration

This function will register a service server at the SSO.

Syntax: /service/registration.ashx

This function will be called by POST method with the following URL-encoded parameters:

* *token*: security code (40 characters long) emitted randomly each time the service server becomes available
* *service*: service name, "slic3r-1.1.7"
* *URL*: URL where the service can be contacted
* *state*: current state of the service in “available”, “loading” and “working”

Return value:

None

Note:

Service servers will publish their state, which will be stored in the service list, every 1 minute.

States can be “available”, “allocated”, “loading” and “working”; every state update will be time-stamped.

“allocated” states newer than 3 minutes won’t be rewritten by an “available” state.

“allocated” states older than 3 minutes will be deleted.

Services older than 2 minutes will be deleted from the service list.

Services list will be stored in database (to allow upgrade with less interruption as possible).

Services list access will be protected by a semaphore mechanism (or equivalent).

Services with URL containing

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter

## Printer interface

### Service request

This function will return an available service (URL and security token), giving priority to the oldest request.

Syntax: /service/request.ashx

This function will be called by POST method with the following URL-encoded parameters:

* *printersn*: printer serial number (see Zim REST web service Printer type function)
* *service*: service description (ex.: “slic3r-1.1.7”)
* *ticket (optional)*: ticket number for subsequent calls

Return value:

Json formatted string (RFC 4627) containing an array:

{ticket: *ticket\_number*

[, URL: *URL*,

token: *service\_security\_token*]}

Note:

Requests will be limited to 1, every 3 seconds, by IP (printers will resend their request every 5 seconds).

Tickets older than 10 seconds will be deleted.

The first available service will be allocated, priority given to the oldest tickets.

If no corresponding service is registered, a 441 error will be returned.

If the ticket number is unknown, a new one will be created and returned.

On allocation, service state will be set “allocated”.

Tickets will be stored in database (to allow upgrade with less interruption as possible).

Tickets and services list access will be protected by a semaphore mechanism (or equivalent).

HTTP error code:

* 432: Missing parameter
* 433: Incorrect parameter
* 435: Too many requests
* 436: Unknown printer
* 441: Unknown service

# Miscellaneous interface

## Alive

This function will request (lightly but effectively) the database and will return an availability code.

Syntax: /alive.ashx

Note:

The login function will be called by POST method with the following URL-encoded parameters:

* *email*
* *password*
* *printersn* : printer serial number (see Zim REST web service Printer type function)
* *printername* : printer user chosen friendly user name

Return value:

A simple text/plain “ok” if everything is all right.

# HTTP error code

* 432: Missing parameter
* 433: Incorrect parameter
* 434: Login failed
* 435: Too many requests
* 436: Unknown printer
* 437: Account already exists
* 438: Unavailable range
* 439: Unknown email
* 440: No redirection
* 441: Service unknown
* 442: Unauthorized user
* 443: Model unknown
* 444: Model already exists
* 445: Print unknown
* 450: File too big