Enetica API Specification

Client Code: Version 1.3
Date: 8th January, 2003

Tale of contents

Enetica API Spec

1. Purpose of document

This document is aimed at those who wish to develop a more in-depth understanding of the API protocols for use with the Enetica wholesale system. It is designed primarily for those users who wish to make extensive modifications to the supplied client code, or to those who want to create a new client (or even API) from scratch. In the case of creating your own API, you will be mostly interested in Section 5 of this document: Server Commands.

The explanations and examples in this document may make use of the Perl language – but there is no limitation to the language you choose to use in connecting to the server.

2. Overview

The purpose of the Enetica API is to support communication between the reseller's web site (client) and the Enetica registration system (server). This allows the client code to issue a full range of commands to the server relating to domain searching (availability lookups, whois information), domain registration and domain management.

The method of this client-server communication is SSL posting. This method provides a reasonable degree of security (through encryption), whilst keeping the method simple and efficient (that is – the client simply makes HTTP requests to the server). In addition to being SSL encrypted, the server program is password protected – and can only be accessed with valid reseller username and password. All commands essentially follow the same steps:

- Client wants to issue a command (eg, domain lookup)
- Client initialises data in the form of name/value pairs for sending to server (eg, “action=lookup” “domain=testsearch.com.au”)
- Client posts data to server, and receives result data in form of name/value pairs (eg, “status=success”) from server (in text format)
- Client parses result from server (placing them in a perl hash in the case of the provided client code) and takes appropriate action

3.API Basic Usage (Perl)

3.1 Usage

The API provided is a perl module (filename: `Enetica/AUClient.pm`), which utilises object orientation to abstract the server operations. A perl CGI-based driver program (filename: `register.cgi`) is also provided to use the features of this module. To use the Perl API from a perl CGI script, you first need to include the API with the perl “use” command. You then need to instantiate the client object with the default connection values defined in the Enetica.conf file. In the register.cgi script provided, these default values are stored in the global %ENETICA hash.

Example:

```
use Enetica::AUClient qw(:default);
$Client = new Enetica::AUClient(%ENETICA);
$Client->login;
```

3.2 Methods

The following methods are available to the Client object:

`login, validate, send_cmd, company_lookup, trademark_lookup`

These are described in greater detail below:

login: This method is used to initialise your login details for connecting to the server. Please note that this operation does not actually connect to the server – but merely sets up a few variables.

Sample Usage:

```
$Client->login;
```

validate: This method is used for validating data input. It returns a hash result which contains an error code and, in some cases, an error message. A hash reference (containing the data to be validated) is passed to this method.

Sample Usage:

```
my %verify_results = $Client->validate(\%data);
unless ($verify_results{is_success}) {
    error_out($verify_results{error_msg});
    exit;
}
```

send_cmd: This is the most important method of the Client object, and is how individual commands are send to the server (from domain lookups to actually registering a domain). The full list of commands that can be issued are detailed in section 5.

Sample Usage:

```
%lookup_results = $Client->send_cmd("lookup",%data);

%results = $Client->send_cmd("register",%data);

...
```

company_lookup: This exists as a separate method of the Client object, though it is possible to simply use the send_cmd directly with the server. It still exists for backward compatibility, but you should use the “send_cmd” format instead.

Sample Usage:

```
my %abn_results = $Client->company_lookup(\%in);
if ($abn_results{status} ne "success") {
    error_out($abn_results{reason});
    exit;
}
```

Or, using send_cmd: `$Client->send_cmd(“company_lookup”, \%hash)`

trademark_lookup: As with the company_lookup method, this method is now defunct. You should instead use the send_cmd method for performing a trademark lookup.

Sample Usage:

```
my %tm_results = $Client->trademark_lookup(\%in);
if ($tm_results{status} ne "success") {
    error_out($tm_results{reason});
    exit;
}
```

4.Encoding and Decoding Messages

Data is sent to and from the server as name/value pairs (via SSL post). Value fields are URL encoded. The “Key” value in each table is the name of the variable being passed. For example, in the case of domain lookups, data is SSL-posted to the enetica server in the following format:

```
action=lookup
domain=enetica.com.au
```

Data would then be returned in the format:

```
status=unavailable
response_code=200
reason=domain%20taken
```

If you are writing your own API, you will need to parse the result data yourself. If you are using the provided client library, then the return values are automatically placed into a perl hash.

5.Server Commands

This section provides the specifications for the commands issued to the server.cgi. Each command is separated into 2 components: Command details and Return data. If using the Perl API provided, the commands in this section are sent via the “send_cmd” method.

Sample Usage:

```
my %renew_results = $Client->send_cmd("check_renewtransfer",%domaindata);
if ($renew_results{renewstatus} eq "success")
{
    ### Do something with return results
}
else
{
    error_out($renew_results{reason});
    exit;
}
```

Usage of the send_cmd method is always of the form:

```
%results = $Client->send_cmd(“command”, %arguments);
```

In the above example:

command:	this is the action to be performed by the server.
%arguments:	this is a hash of values (command arguments) sent to the server.
%results:	this is a hash of return values sent by the server.

When reading through this section, the “Command details” table contains the value of the *arguments* hash (with the “action” key corresponding to *command*), and the “Return data” table documents the elements of the *results* hash (returned by the server).

5.1 Domain Lookup: lookup

Performs a lookup on the availability of a domain name. For .au domains, this feature should not be required – as the source code is distributed with the “aulookup” utility (which is much faster, and allows bulk searches).

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“lookup”
domain	Valid domain string	Yes	xyz.com.au, xyz.com

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“available”, “unavailable”
response_code	number	always	200
reason	string	if unavailable	“reserved word or phrase”, “domain taken”

5.2 ASIC/Company Verification: company_lookup

This action is used to determine if a provided ABN, ACN or state business number is valid by doing a search on the ASIC site. If the company details are found, it returns the company name (and in the case of a company with multiple trading names, returns these as well).

Please note that if the business number is not found that this does not necessarily mean the number is invalid (for example, the ASIC website may have been down).

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	"company_lookup"
abn	number (ABN or ACN)	No	39 087 987 988
brn_number	string	If no abn or acn	BN999999
brn_state	string	If brn	NSW
tradingascheck	boolean	no	"1" - used only if provided both abn and brn

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	"sucess", "failed"
abn_lookup_failed	number	If failed	1
reason	string	if failed	"invalid number"
abn_acn_name	string	If found	Eg: "ENETICA PTY LTD"
tradingname_list	list (comma delimited)	If abn and alternative names listed.	Note: Many companies don't have additional trading names.
brn_name	string	If found	

5.3 Trademark Verification: trademark_lookup

Similar to company lookup – but does a search on trademark number. Returns the trademark name if found. This is only needed if domain applicants are basing their domain eligibility on a trademark.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	"trademark_lookup"
trademark_number	number	Yes	111111

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	"sucess", "failed"
reason	string	if failed	"invalid number"
name	string	If found	Eg: "ALTRO"

5.4 Check Renew/Transfer status: check_renewtransfer

Used for .au domains to determine whether a renewal is coming from another registrar. If it is, then order type of "transfer" is returned. If it isn't, then type is "renew". If type is "renew", then the domain can only be renewed within 90 days of expiry. Transfers on the other hand can occur at any time.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	"check_renewtransfer"
domain	string (valid domain)	Yes	xyz.com.au
password	string	If transfer	Domain password, for example: A012345

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	"sucess", "failed"
reason	string	if failed	"Missing Domain", "Domain not due for renewal"
renewstatus	string	always	"success", "failed"
ordertype	string	If status = "success" AND domain is .au domain.	"renew" or "transfer"
days	number	If status = "success" AND ordertype = "renew" AND domain is .au.	Number of days until the domain expires, eg: 42

If for example you try to renew a domain which is already under the Enetica system, but which isn't due to expire for another 100 days, you would receive the following data:

```
status=failed
renewstatus=failed
reason=Domain not due for renewal.  Please try again in 10 days
days=100
```

Alternativly, if you try issue a "check_renewtransfer" command when there is only 89 days until the domain expires, then you would receive the following:

```
status=success
renewstatus=success
ordertype=renew
days=89
```

Both of the above cases assumed that the domain was already registered through Enetica. If the domain is presently registered under another registrar, then in EITHER of the above cases, you would receive the result (as long as you provided the correct password):

```
status=success
renewstatus=sucess
ordertype=transfer
```

Or, if the password provided was incorrect:

```
status=failed
renewstatus=failed
reason=Password required for domain transfer.  Please visit auDA to retrieve your password.
```

5.5 Register Domain: register

This is the most important command you send to the server – and allows you to place a domain registration order in the pending queue, which you can then view through the RWI (Reseller's Web Interface). If you wish to process/register the domain straight away, you can then issue the “process” command. If you do not send the process command from your client code, the application will remain in your pending queue until you either process or cancel it from the RWI.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“register”
domain	string (comma delimited list of valid domain)	Yes	xyz.com.au,xyz2.com.au
ordertype	string	Yes	“new”, “renew”, “transfer”, etc.
password	string	If transfer	A012345
num_years	number	If gTLD	Number of years for registration (between 1 and 10 for gTLD's)
claim_type	number	If .au	1 to 11 1 = Exact match 2 = Abbreviation 3 = Acronym 4 = Refers to product 5 = Program we administer 6 = Refers to service 7 = Event we sponsor 8 = Activity we teach/train 9 = Venue we operate 10 = Name of profession practiced by our employees 11 = Derived from real name (for id.au only)
abn	number (valid ACN or ABN)	Sometimes* (see footnote at end of table)	ABN or can for company registering .au domain
company	string	If .”abn”	“Enetica Pty Ltd”
trademark_number	number	Sometimes*	Only required if claim based on trademark
trademark_name	string	Sometimes*	Only required if claim based on trademark
brn	string	Sometimes*	Only required if claim based on trading name
trading_name	string	Sometimes*	Only required if claim based on trading name

Key	***Type***	***Required***	***Value/Examples***
business_type	string	If .au	*One of the following:* ACN ABN VIC BN NSW BN SA BN NT BN WA BN TAS BN ACT BN QLD BN OTHER
elig_type	string	If .au. This field is required for .au domains to determine the eligibility criteria for domain.	*One of the following:* Company Registered Business Sole Trader Trademark Owner Pending TM Owner Incorporated Association Club Non-profit Organisation Charity Trade Union Industry Body Commercial Statutory Body Religious/Church Group Political Party Other **Note:** Not all of the above elig_types are allowed for all domain types. Refer to .au policy for more information
connection	text/string	Optional (not for gTLD)	This field was previously used for “close and substantial connection” - but is now essentially a “notes” field. It is to be used when the registrant needs to provide additional information that may assist with their application review.
url_forwarding	boolean (0 or 1)	No (default 0)	Whether domain is going to use URL forwarding.
mail_forwarding	boolean (0 or 1)	No (default 0)	Whether to enable mail forwarding (incurs cost for reseller)

Key	*Type*	*Required*	*Value/Examples*
spam_forwarding	boolean (0 or 1)	No (default 0). Ignored if mail forwarding = 0	Whether to enable spam filtering on mail forwarding (if enabled). Incurs fee.
Contact Details: owner_first_name, owner_last_name, owner_org_name, owner_address1, owner_address2, owner_city, owner_postcode, owner_state, owner_country, owner_phone, owner_fax, owner_email	strings	Always (some fields, such as fax and address2 are optional).	Self-explanatory
tech_same	boolean	No (default 0)	Set to 1 to copy tech contact details from owner details.
billing_same	boolean	No (default 0)	Set to 1 to copy billing details from owner details.
Tech Contact: tech_first_name ...	strings	If tech_same is set to 0.	Self-explanatory
Billing Contact: billing_first_name ...	strings	If billing_same is set to 0.	Self-explanatory

*Sometimes: These fields are never required for gTLD's, and are only sometimes required for .au applications. Please refer to the .au allocation policy for more information on when this information is required.

Return data:

Key	*Type*	*When*	*Value/Examples*
status	string	always	"sucess", "failed"
reason	string	if failed	"Missing Domain",
domain1, domain2, ...	strings	If status=success	Each domain passed to server is returned with unique id...
status1, status2, ...	strings	If status = "success".	Separate status result for each domain (eg, status2 gives result for domain2)
reason1, reason2, reason3, ...	strimgs	reasonX is provided for each statusX that has value of "failed"	Separate reason provided for each failed domain.
id1, id2, id3, ...	numbers	For each success	unique order for each domain: id1=20997

Please note that with the return data from the "register" command, that you will need to save the "id" values returned by the server if you wish to process them immediately (ie, from the client code). Each id field (ie, id1, id2, id3 ...) corresponds to each domain you sent to the server – and is a unique order id that applies ONLY to that domain. For more information on processing orders (once they have been pended by the "register" command), see section 5.6, below.

5.6 Process Registration: process

Submits a domain from the "pending" queue to the "progress" queue. If the domain application has already been reviewed (and approved) by our staff, then domain is registered immediately and the status changed to "processed".

To process a previously pended application from the client code, you need to pass the unique order id for each domain up to the server. These order id's are provided by the server when the order is pended.

Command details:

Key	*Type*	*Required*	*Value/Examples*
action	string	Yes	“process”
ids	list of order-id's (comma delimited list of numbers)	Yes	xyz.com.au

Return data:

Key	*Type*	*When*	*Value/Examples*
statusXX, ...	string	always	XX refers to order_id. A separate status is returned for each order id passed to the server.
reasonXX ...	strings	For	Eg, “Order does not exist”
idXX ...	numbers	if success	A separate return code is sent back for each successfully processed order. These values include: 0 – domain processed 300 – no matching order id 400 – not enough credits (order remains in pending queue)

5.7 Log into domain management: login

Creates a session cookie for users to manage their domain. This cookie needs to be passed back up to the server every time you issue a domain management command (along with domain name).

Command details:

Key	*Type*	*Required*	*Value/Examples*
action	string	Yes	“login”
domain	string (valid domain)	Yes	xyz.com.au
password	string	Yes	A05678

Return data:

Key	*Type*	*When*	*Value/Examples*
status	string	always	“sucess”, “failed”
reason	string	if failed	Eg, “Authorisation Failed”
cookie	string	if success	5XcZgavqF8bhOZ9lP24TsJkiq

5.8 Log out of domain management: logout

Logs out of domain management (deletes the cookie). Requires cookie (ie, user must have previously logged in).

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“logout”
domain	string (valid domain)	Yes	xyz.com.au
id	String (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	if failed	Eg, “Invalid Cookie”

5.9 Retrieve Domain Info: get_domain_info

Retrieves domain information as name-value pairs. This is used primarily to get contact and nameserver info (used for domain management). Requires cookie (ie, user must already be logged in)

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“get_domain_info”
domain	string (valid domain)	Yes	xyz.com.au
id	String (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
info	string	if success	Pre-formatted domain info
suggested_names	string (Note: this key may exist multiple times)	if suggestions found	* This is a list of suggested (and available) domain names that the owner might be interested in registering (based on domain info).

5.10 Retrieve Nameserver Info: get_nameserver_info

Retrieves the nameserver details for the domain. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	"get_domain_info"
domain	string (valid domain)	Yes	xyz.com.au
id	String (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	"sucess", "failed"
reason	string	If failed	"Invalid Cookie"

ns0, ns1, ...	string	if found	ns0=ns1.enetica.com.au
			ns1=ns2.enetica.com.au
			...

5.11 Remove Nameservers: delete_nameservers

Drops nameservers for a domain. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	"delete_nameservers"
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
ns	Comma delimited list (strings)	Yes	ns=ns1.enetica.com.au

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	"sucess", "failed"
reason	string	If failed	"Invalid Cookie"
message	string	if success	"Nameservers Deleted"

5.12 Add Nameservers: add_nameservers

Adds nameservers to a domain. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“add_nameservers”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
ns	Comma delimited list (strings)	Yes	ns=ns1.enetica.com.au

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”
message	string	if success	“Nameservers Created”

5.13 Retrieve Contact Info: get_contact_info

Retrieves contact information (admin, tech, billing) for a domain. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“get_contact_info”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”
admin_id admin_name admin_email admin_organisation admin_address1 admin_address2 admin_city admin_postcode admin_state admin_country admin_phone admin_fax admin_selected	strings	if success	Contact details... Note: contact info is provided for admin, tech and billing (ie, admin_name, tech_name, billing_name, etc).

5.14 Add Domain Contact: add_contact

Adds contact details for a domain. Requires cookie.

Command details:

Key	*Type*	*Required*	*Value/Examples*
action	string	Yes	“add_contact”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
type	string	Yes	“admin”, “tech” or “billing”
admin_name admin_email admin_organisation admin_address1 admin_address2 admin_city admin_postcode admin_state admin_country admin_phone admin_fax	strings	Yes	Note: contact info is provided for admin, tech and billing (ie, admin_name, tech_name, billing_name, etc).

Return data:

Key	*Type*	*When*	*Value/Examples*
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”
contact_id	strings	if success	New contact id created for contact (eg, “KF9999”).

5.15 Remove Domain Contact: drop_contact

Drops a contact from a domain. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“drop_contact”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
type	string	Yes	“admin”, “tech” or “billing”
dropid	string	Yes	Contact id to drop (eg KF9999).

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”

5.16 Retrieve URL Forwarding details: get_url_forwarding

If URL forwarding is enabled, this returns the details (URL, title, keywords). Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“get_url_forwarding”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”
notfound	Number (0, 1)	If success	0, if url forwarding entry exists 1, if url entry is not found
dest_url	string	If notfound=0	Destination URL Eg, http://www.enetica.com.au/
title	string	If notfound=0	Web page title
keywords	string	If notfound=0	Meta tags: Web page keywords
description	string	If notfound=0	Meta tags: Web page description

5.17 Update URL Forwarding: update_url_forwarding

Updates URL forwarding details. Please note that in order for URL or email forwarding to work, the domain must be delegated to the Enetica nameservers. Requires cookie.

Command details:

Key	*Type*	*Required*	*Value/Examples*
action	string	Yes	“update_url_forwarding”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
dest_url	string	Yes	Destination URL Eg, http://www.enetica.com.au/
title	string	Optional	Web page title
keywords	string	Optional	Meta tags: Web page keywords
description	string	Optional	Meta tags: Web page description

Return data:

Key	*Type*	*When*	*Value/Examples*
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”, “URL Forwarding not enabled”

5.18 Retrieve Email Forwarding details: get_email_forwarding

If email forwarding is enabled, this returns forwarding details. Requires cookie.

Command details:

Key	*Type*	*Required*	*Value/Examples*
action	string	Yes	“get_email_forwarding”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	*Type*	*When*	*Value/Examples*
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”
mail	Number (0, 1)	If success	1 if mail forwarding is enabled 0 if it is not
spam	string	If success	1 if spam filtering is enabled 0 if it is not.
spam_address	string	If spam = 1	Eg, spam@hotmail.com
name1 ... name5	strings	If mail = 1 and value exists	Username portion of email address, eg “fred”, “joe.bloggs”
address1 ... address5	strings	If mail = 1 and value exists	Email addresses that above users forward to, eg “support@enetica.com.au”

5.19 Update Email Forwarding: update_email_forwarding

Updates email forwarding details. Please note that in order for URL or email forwarding to work, the domain must be delegated to the Enetica nameservers. Requires cookie.

<u>Note</u>: This will delete all previous email forwarding entries for this domain, befor adding new values. This means that if you wish to simply add one address, that you need to re-send all details.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“update_email_forwarding”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
spam_address	string	If spam filtering is enabled.	Eg, spam@hotmail.com Field is ignored if spam forwarding not enabled for domain.
name1 ... name5	strings	Yes (at least one forwarding account)	Username portion of email address, eg “fred” or “joe.bloggs”
address1 ... address5	strings	Yes (ast least one forwarding address)	Email addresses that above users forward to, eg “support@enetica.com.au”

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”, “Mail Forwarding not enabled”
mail	boolean	always	1 if enabled, 0 otherwise
spam	boolean	always	1 if enabled, 0 otherwise

5.20 Retrieve Domain Certificate: view_domain_cert

Retrieves formatted domain certificate for a domain registered through our system. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“view_domain_cert”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”
cert	string	if success	Pre-formatted domain certificate (HTML).

5.21 Modify Password: update_password

Allows a logged in user to change the password for their domain. Requires cookie.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“update_password”
domain	string (valid domain)	Yes	xyz.com.au
id	string (valid cookie)	Yes	5XcZgavqF8bhOZ9lP24TsJkiq
old_pass	string	Yes	“A012345”
new_pass1	string	Yes	“secret123”
new_pass2	string	Yes	“secret123”

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”, “Password mismatch”
message	string	If success	“Password Updated”

5.22 Email password to domain owner: mail_pass

Sends the password for a domain to the registered domain contact.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“mail_pass”
domain	string (valid domain)	Yes	xyz.com.au

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Domain not under our management”
email	string	If success	Email address to which the password was send (the domain registrant's email address)

5.23 Update IP: update_ip

Adds/updates an IP address for an already existing nameserver.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“update_password”
domain	string (valid domain)	Yes	xyz.com.au
ns	string (comma delimited list of ns:ip)	Yes	List is comma delimited, and each element in list contains nameserver-colon-IP. eg: ns1.xyz.com:124.124.124.124

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Cookie”

5.24 Domain Whois: whois_domain

Retrieves formatted domain info for an existing .au domain.

Command details:

Key	***Type***	***Required***	***Value/Examples***
action	string	Yes	“whois_domain”
domain	string (valid domain)	Yes	xyz.com.au

Return data:

Key	***Type***	***When***	***Value/Examples***
status	string	always	“sucess”, “failed”
reason	string	If failed	“Invalid Domain”
whois	string	if success	Pre-formatted whois output (Text).