Online Domain Tools

API Specification

Table of Contents

Support contact: http://support.online-domain-tools.com/

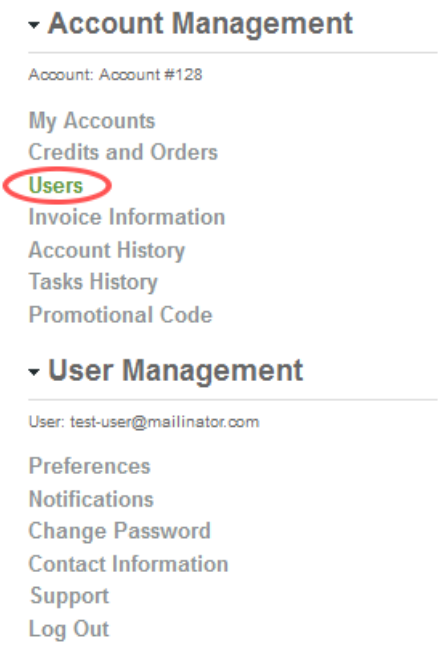
Last update: November 27, 2014

1 API Credentials

Access to API is provided using

- API Key – Generated key provided to partner (e.g. ODT-API-XXX), and
- API Secret – Generated password used for authentication (random string).

Before API can be used, the user has to register an ODT account and obtain API credentials. When you are logged in Online Domain Tools, click Users link in the right side menu under Account Management.



Click **Edit** to edit the user, which you want to enable API access for. The user has to be a fully privileged user, so we recommend you enable API access for the account owner (the user that created the account).

On the next screen, click **Enable** to enable the API access and see the API credentials – API Key and API Secret.

2 API General Mechanisms

2.1 Authentication

All calls to ODT API are authenticated HTTPS POST requests. All calls are made to URL in a generic format:

```
https://secured.online-domain-tools.com/api/user/$ACTION/
```

Where $ACTION is specific to the request operation.

The authentication is implemented using three HTTP headers:

- `Key` – API Key.
- `Sign` – Lower case hex representation of HMAC SHA512 of the request.
- `Time` – Source server UTC date time specification in format YYYY-MM-DD hh:mm:ss. 24-hour format with leading zeros is required.

`Time` is considered valid, if it is not off by more than 15 minutes compared to our server time.

`Sign` is a hash calculated from a concatenation of `Key`, `Time` and request `Body` using partner's API secret:

```
$SIGN := hmac_sha512($SECRET, $KEY + $TIME + $BODY).tolower()
```

See below section 10 for a working sample code.

2.2 Calling Modes

Each API call can operate either in synchronous mode, asynchronous mode, polling mode, or any combination of these modes.

2.2.1 Synchronous Mode

The default mode is synchronous mode and API calls that support synchronous mode do not need to specify extra parameters to use synchronous mode. In synchronous mode, the API server replies immediately with the final result.

2.2.2 Asynchronous Mode

API calls that support asynchronous mode can be called with extra POST parameter `asyncCallback` that forces the call to run in asynchronous mode.

- POST body arguments:
  - `asyncCallback` *(optional)* – If present, it forces the API call to be processed in asynchronous mode and it must be set to HTTP or HTTPS URL that is called by ODT API server once the operation completes. The callback URL is called only in case the return value `success` is 1 (see section 2.3) in the call API response.

The target script specified in the `asyncCallback` argument receives a POST request with Content-Type header set to `application/json`.

The callback processing code must finish as soon as possible and send the response to the server callback request that contains a single text line:

```
ODT: OK
```

Callback targets that repeatedly do not respond with this line within 5 seconds can be blacklisted.

Many API calls support `testMode` parameter for development purposes. If both `testMode` and `asyncCallback` arguments are set, validation is performed as in synchronous mode, but callback URL is not called.

2.2.3 Polling Mode

API calls that support polling mode can be called with extra POST parameter `polling` that forces the call to run in polling mode.

- POST body arguments:
  - `polling` *(optional)* – If set to 1, it forces the API call to be processed in polling mode. In this mode, the API server replies immediately with a response, which informs the caller that the operation is in progress and provides a URL that the caller is expected to check periodically to obtain the operation result. The result URL is provided in return value `resultUrl`, and it is provided only if the value `success` is 1 (see section 2.3) in the call API response. See section 2.3.1 below for more information.

Many API calls support `testMode` parameter for development purposes. If both `testMode` and `polling` arguments are set, validation is performed as in synchronous mode, but no target result URL is provided and the operation is not executed.

The documentation of each API that supports polling mode contains information about the reasonable

frequency of polling. Callers polling with much higher frequencies can be temporarily banned.

The caller can access the `resultUrl` using both GET and POST methods, the authentication is not required in this case.

2.3 Response

A generic response to a Partner API request is in JSON format.

In case of a successful call, the format is

```
{
  "success":1,
  ...
}
```

In case of an error, the format is

```
{
  "success":0,
  "message":$MESSAGE
}
```

where `$MESSAGE` is a string.

2.3.1 Polling Response

In polling mode, the response format of a successful call is

```
{
  "success":1,
  "resultUrl":"https://...",
  ...
}
```

The `resultUrl` is the URL that the caller is expected to poll to obtain the operation result. If the result is not available when the caller checks, the value `success` is `0` and the error `message` is set to `Pending`. All other values means that the task is completed (successfully or not) and the caller should not check the `resultUrl` anymore. Callers that keep polling after the task is complete can be blacklisted.

2.3.2 General Errors

API calls can generate following general errors:

Value of returned `$MESSAGE`	Description
`POST method is required.`	The request used other than HTTP POST method.
`Authentication failed. Key header is missing.`	The request did not contain HTTP header `Key`.
`Authentication failed. Sign header is missing.`	The request did not contain HTTP header `Sign`.
`Authentication failed. Time header is missing.`	The request did not contain HTTP header `Time`.
`Authentication failed. Invalid signature.`	The signature was invalid.
`Authentication failed. Invalid time. Server time is YYYY-MM-DD hh:mm:ss.`	The value of `Time` header was invalid. Either the time was sent in an invalid format, or the provided time differed for more than 15 minutes compared to our server's UTC time. Our server's time in UTC is returned in the

	response.
`Synchronous mode is not supported.`	The API call does not support synchronous mode.
`Asynchronous mode is not supported.`	The API call does not support asynchronous mode.
`Polling mode is not supported.`	The API call does not support polling mode.
`Insufficient funds. Call required X, balance is Y.`	The API call required X credits, but the account, in which context was the call made, only has Y credits. If this error is returned, an automatic email notification is sent to the client's email address in order to inform him about this problem. The client receives no more than 1 notification every 24 hours.
`Access denied.`	The API was called by a user who does not have Spend credits right to the account.
`Blacklisted.`	The caller does not conform to the API specification and used IP or URL was blacklisted.
`Slow down.`	The caller executes too many requests too quickly.
`Pending.`	The polling mode result URL informs the caller that the results are not available yet and it should keep polling.
`Invalid argument. ARG is invalid.`	The value of `ARG` argument is invalid. The argument's value was either in invalid format, too long, or not unique as required.
`Invalid argument. ARG is missing.`	The request did not contain `ARG` argument, which was required.
`Service error.`	The request could not be completed due to error in the service. This could happen when a service is in a maintenance mode or due to an unexpected error. Please try again in a few minutes. If the problem persists for more than 30 minutes, please contact support.

3 Account API Calls

3.1 Authentication Test

Supported modes: synchronous only

This action is implemented for the purpose of testing the authentication mechanism.

- $ACTION – account/authTest
- POST body arguments are optional and can be arbitrary. A developer can send any POST arguments in order to tests an implementation of the authentication mechanism.

3.1.1 Success

No additional values are returned.

3.1.2 Failure

Only general errors are returned.

3.2 Account Information

Supported modes: synchronous only

Returns basic information about the account.

- $ACTION – account/info
- No POST body arguments.

3.2.1 Success

Following values are returned:

Name	Type	Description
`name`	string	Name of the account.
`owner`	string	Email address of the account owner.
`creditsWallet`	double	Current balance of account's wallet.
`creditsDaily`	double	Current balance of account's daily credits.
`creditsDailyMax`	double	Maximal number of account's daily credits.

3.2.2 Failure

Only general errors are returned.

4 Bulk Email Verifier API Calls

Tool webpage: http://email-verifier.online-domain-tools.com/

4.1 Check

Supported modes: asynchronous + polling (5 seconds)

Checks whether emails on input are valid and provides additional information about the email if available.

- $ACTION – tool/email-verifier/check
- POST body arguments:
  - `testMode` *(optional)* – If set to 1, a successful API call does not perform an email validation. It just validates the input and checks that the caller is able to perform the call (e.g. has enough credits). Returned response may be incomplete.
  - `emails` – Comma-separated list of emails to validate.
  - `greylistingDelay` *(optional)* – How long should the tool wait when a greylisting is detected, before trying to verify the email address again. Specified as integer number of seconds between 0 and 600. If set to 0, greylisting rechecking is disabled. The default value is 0.

4.1.1 Success

Following values are returned:

Name	Type	Description
`toolName`	string	Name of the tool.
`status`	struct	`TOOL_STATUS` structure, see section 9 below.
`output`	struct	`EMAIL_VERIFIER_OUTPUT` structure, see below.

`EMAIL_VERIFIER_OUTPUT` structure is defined as follows:

Name	Type	Description
`stats`	struct	`EMAIL_VERIFIER_STATS` structure, see below.
`emailsInfo`	array	Array of `EMAIL_VERIFIER_EMAIL_INFO` structures, see below.

`EMAIL_VERIFIER_STATS` structure is defined as follows:

Name	Type	Description
`invalidFormatCount`	int	Total number of input entries that do not represent an email address.
`validCount`	int	Number of valid emails.
`invalidCount`	int	Number of invalid emails.
`catchAllCount`	int	Number of emails on domains with catch-all address enabled.
`disposableCount`	int	Number of emails on domains hosted by disposable services.

`EMAIL_VERIFIER_EMAIL_INFO` structure is defined as follows:

Name	Type	Description

email	string	Email address from input.
isInvalidFormat	bool	true if email does not represent an email address, false otherwise.
isValid	bool	true if email address is valid, false otherwise.
isCatchAll	bool	true if email is on domain with catch-all address enabled, false otherwise.
isDisposable	bool	true if email is on domain hosted by disposable service, false otherwise.
finalStatus	int	One of EMAIL_VERIFIER_EMAIL_STATUS values greater than 1, see below.
progressLog	array	Array of EMAIL_VERIFIER_PROGRESS_INFO structures, see below.

EMAIL_VERIFIER_EMAIL_STATUS enumeration is defined as follows:

Value	Name	Description
0	Unknown	No check has been done.
1	GreylistingWait	Validation still in progress. Possible greylisting detected, waiting before next attempt.
2	Ok	This email address is valid. Message to this address will be delivered.
3	OkLikely	Most likely, this email is valid. Message to this address will likely be delivered. Greylisting mechanism is probably implemented on this address.
4	UnknownWillDeliver	This email address is valid. However, every email address on its domain is considered as valid. It is thus unknown whether a message to this email address will be actually delivered to a real recipient.
5	UnknownWillNotDeliver	It is unknown whether this email is valid or not. However, the current state of its mail servers will cause that a message to this email will not be delivered.
6	Bad	This email address is invalid. Message to this address will not be delivered.

EMAIL_VERIFIER_PROGRESS_INFO structure is defined as follows:

Name	Type	Description
progressStatus	int	One of EMAIL_VERIFIER_PROGRESS_STATUS values, see below.
emailStatus	int	One of EMAIL_VERIFIER_EMAIL_STATUS values, see above.
timestamp	string	UTC date time specification of when did the event occur. Format is YYYY-MM-DD hh:mm:ss, 24-hour format with leading zeros.
description	string	Description of the event.

EMAIL_VERIFIER_PROGRESS_STATUS enumeration is defined as follows:

Value	Name	Description

0	`Init`	Initialization phase of verification process.
1	`MxRecord`	Obtaining list of mail servers from DNS.
2	`SmtpConnection`	Connecting to mail server.
3	`SmtpGreeting`	Server must send greeting.
4	`SmtpHelo`	EHLO command in progress.
5	`SmtpMailFrom`	MAIL FROM command in progress.
6	`SmtpCatchAllRcptTo`	Trying invalid email address to detect catch-all address settings.
7	`SmtpRcptTo`	Verifying email validity using RCPT TO command.
8	`SmtpGreyListingCheck`	Handling possible greylisting.
9	`Finished`	Verification process is finished.

4.1.2 Failure

Only general errors are returned.

5 Password Checker Online API Calls

Tool webpage: http://password-checker.online-domain-tools.com/

5.1 DictionaryCheck

Supported modes: synchronous

Performs dictionary check of a password.

- $ACTION – tool/password-checker/dictionary-check
- POST body arguments:
  - `testMode` *(optional)* – If set to 1, a successful API call does not perform a check. It just validates the input and checks that the caller is able to perform the call (e.g. has enough credits). Returned response may be incomplete.
  - `password` – Password to be evaluated. Maximum length is 128 characters.

5.1.1 Success

Following values are returned:

Name	Type	Description
`toolName`	string	Name of the tool.
`status`	struct	`TOOL_STATUS` structure, see section 9 below.
`safe`	bool	Indicates whether or not the password passes the dictionary analysis.
`message`	string	Optionally, if `safe` is `false`, this value is filled with additional details explaining why the password is considered as unsafe.

5.1.2 Failure

Only general errors are returned.

6 Online Mail Server Blacklist Checker API Calls

Tool webpage: http://mail-blacklist-checker.online-domain-tools.com/

6.1 Check

Supported modes: synchronous + asynchronous + polling (5 seconds)

Tests whether a specific machine is listed in one or more mail server blacklists.

- $ACTION – tool/blacklist-checker/check
- POST body arguments:
  - `testMode` *(optional)* – If set to 1, a successful API call does not perform a blacklist check. It just validates the input and checks that the caller is able to perform the call (e.g. has enough credits). Returned response may be incomplete.
  - `target` – Any valid host name or IPv4 address except for localhost and special-use addresses. If a host name is used instead of IPv4 address, it is translated to IPv4 address before the check is made.

6.1.1 Success

Following values are returned:

Name	Type	Description
`toolName`	string	Name of the tool.
`status`	struct	`TOOL_STATUS` structure, see section 9 below.
`output`	struct	`BLACKLIST_CHECKER_OUTPUT` structure, see below.

`BLACKLIST_CHECKER_OUTPUT` structure is defined as follows:

Name	Type	Description
`stats`	struct	`BLACKLIST_CHECKER_STATS` structure, see below.
`blacklisted`	array	Array of strings. Each string represents a name of a blacklist on which the target host is blacklisted. If the target host is not blacklisted on any list, the list is empty.
`blacklists`	array	Array of `BLACKLIST_CHECKER_BLACKLIST` structures, see below.

`BLACKLIST_CHECKER_STATS` structure is defined as follows:

Name	Type	Description
`blacklistsCount`	int	Total number of blacklists that the target host was checked on.
`blacklistedCount`	int	Number of blacklists that list the target host.
`okCount`	int	Number of blacklists that do not list the target host.
`naCount`	int	Number of blacklists that did not respond.

`BLACKLIST_CHECKER_BLACKLIST` structure is defined as follows:

Name	Type	Description
`host`	string	Name of the blacklist.
`status`	string	One of the following values: • `ok` – The blacklist does not list the target host. • `listed` – The blacklist lists the target host. Additional information may be provided in `reason`. • `n/a` – The blacklist did not respond.
`reason`	string	Optionally, if `status` is `listed` and the specific blacklist provided this information, additional details about the listing. In this value, some blacklists provide information about how to ask for unlisting, or where to get more information.

6.1.2 Failure

Only general errors are returned.

7 Online Website Link Checker API Calls

Tool webpage: http://website-link-checker.online-domain-tools.com/

7.1 Check

Supported modes: asynchronous + polling (10 seconds)

Crawls a target URL and checks all its links in HTML and CSS codes.

- $ACTION – tool/website-link-checker/check
- POST body arguments:
  - `testMode` *(optional)* – If set to 1, a successful API call does not perform a link check. It just validates the input and checks that the caller is able to perform the call (e.g. has enough credits). Returned response may be incomplete.
  - `url` – URL address, on which the link checker should start looking for links.
  - `depth` – Maximal crawling depth – a number between 1 and 10. See the tool's Usage section for more information.
  - `range` – Specifies which webpages should be checked for (broken) links. One of the following values:
    - `wholeDomain` – The link checker will crawl across all subdomains of the URL's main domain.
    - `subdomainOnly` – Forces the link checker to check only pages within a single subdomain that is specified in the `url` parameter. Note that `www.domain.tld` and `domain.tld` are always considered as the same subdomain.
  - `pageLimit` – Maximum number of pages to crawl – a number between 1 and 10000. This option affects the pricing. See the tool's Usage section for more information.
  - `checkForms` – If set to `1`, the link checker will check links defined by HTML forms – `action` and `formaction` attributes. If set to `0`, form links will not be checked.
  - `checkCss` – If set to `1`, the link checker will check links in CSS files. If set to `0`, links in CSS files will not be checked.

- `respectRobots` – If set to `1`, the link checker will respect `robots.txt` protocol. If set to `0`, `robots.txt` protocol will be ignored. This option affects the pricing.
- `brokenLinksOnly` – If set to `1`, the output will contain only broken links. If set to `0`, the output will contain all links.

7.1.1 Success

Following values are returned:

Name	Type	Description
`toolName`	string	Name of the tool.
`status`	struct	`TOOL_STATUS` structure, see section 9 below.
`output`	struct	`WEBSITE_LINK_CHECKER_OUTPUT` structure, see below.

`WEBSITE_LINK_CHECKER_OUTPUT` structure is defined as follows:

Name	Type	Description
`stats`	struct	`WEBSITE_LINK_CHECKER_STATS` structure, see below.
`brokenPages`	array	Array of `WEBSITE_LINK_CHECKER_PAGE` structures, see below. Each structure represents a web page with one or more broken links.
`workingPages`	array	Array of `WEBSITE_LINK_CHECKER_PAGE` structures, see below. Each structure represents a web page with one or more working links.

`WEBSITE_LINK_CHECKER_STATS` structure is defined as follows:

Name	Type	Description
`processedLinksCount`	int	Total number of checked links.
`brokenPages`	int	Number of pages in `brokenPages` array.
`workingPages`	int	Number of pages in `workingPages` array.

`WEBSITE_LINK_CHECKER_PAGE` structure is defined as follows:

Name	Type	Description
`pageUrl`	string	Name of the blacklist.
`links`	array	Array of `WEBSITE_LINK_CHECKER_LINK` structures, see below. Each structure represents a link on the web page with `pageUrl` address. In case of members of brokenPages array, a link structure represents a broken link. In case of members of workingPages array, a link structure represents a working link.

`WEBSITE_LINK_CHECKER_LINK` structure is defined as follows:

Name	Type	Description
`type`	string	One of the following values: • `ok` – The link is not broken and the target page can be crawled and analyzed. • `error` – The link is broken.

		• `external` – The link is not broken but the target page can not be analyzed because the link is outside the specified crawling range. • `forbidden` – The link was not checked because it is disallowed by `robots.txt` protocol.
`statusCode`	string	HTTP status code or error. In case the link is redirected, this is the status code of the final page in the redirection chain. To get HTTP status code of the first redirect, see `redirectCode`. To get HTTP status codes of other pages in the redirection chain, see `redirects`. Possible values are: • *`NNN`* – Three digit HTTP status code (such as 200 or 404). These codes are returned by target HTTP servers. • `ProtocolError` – Error in HTTP protocol. • `ConnectFailure` – Problem to connect server on the target address. • `ReceiveFailure` – Unable to receive data from the target server. • `RequestCanceled` – Request was canceled because too many redirections occurred. • `NameResolutionFailure` – Unable to resolve target URL's domain.
`contentType`	string	The MIME type of the body of the request.
`url`	string	Full URL of the target page. In case the link is redirected, this is the URL of the first redirect.
`htmlHref`	string	Substring of the target page's HTML or CSS code that was identified as this link. For example, in HTML page, if `a` element is found, `htmlHref` contains the value of its `href` attribute.
`position`	int	Character offset in the analyzed page code, where the `htmlHref` string was found. `position` is set to `0` for the source URL.
`redirectCode`	string	Optionally, if the link is redirected, the original HTTP status code.
`finalUrl`	string	Optionally, if the link is redirected, the final URL of the redirection chain.
`redirects`	array	Optionally, if the link is redirected, an array of the final URL of `WEBSITE_LINK_CHECKER_REDIRECT` structures, see below. Each structure represents an item in the redirection chain.

`WEBSITE_LINK_CHECKER_REDIRECT` structure is defined as follows:

Name	Type	Description
`statusCode`	string	HTTP status code or error. For possible values, see `statusCode` in `WEBSITE_LINK_CHECKER_LINK`.
`url`	array	Full URL of the page.

7.1.2 Failure

Only general errors are returned.

8 Whois Online API Calls

Tool webpage: http://whois.online-domain-tools.com/

8.1 Query

Supported modes: synchronous + asynchronous + polling (5 seconds)

Gets WHOIS information for a domain or an IP address.

- $ACTION – tool/whois/query
- POST body arguments:
  - `testMode` *(optional)* – If set to 1, a successful API call does not perform a WHOIS query. It just validates the input and checks that the caller is able to perform the call (e.g. has enough credits). Returned response may be incomplete.
  - `query` – A second-level domain name for all top-level domains that allow Second-level domain registration such as .com, .net, .info, .org, .eu, .de, etc. Or, a third-level domain for all other top-level domains such as .uk, .br, .au, etc. Or, an IPv4 address. No subdomains, such as www, are allowed.

8.1.1 Success

Following values are returned:

Name	Type	Description
`toolName`	string	Name of the tool.
`status`	struct	`TOOL_STATUS` structure, see section 9 below.
`output`	struct	`WHOIS_OUTPUT` structure, see below.
`rawOutput`	array	Array of strings. Each string represents a complete response from WHOIS server. To get a final WHOIS record, 1-3 WHOIS requests are usually needed. This array contains responses to all requests.

`WHOIS_OUTPUT` structure is defined as follows. However, there is no single standard for WHOIS records format. Each WHOIS response is parsed and as many values as possible are extracted and provided in this structure, but structures for different domains can contain and miss different values.

Name	Type	Description
`domain`	string	Name of the domain described by the record.
`domain_id`	string	Unique domain ID used by some registrars.
`status`	array	Array of strings that describe the status of the domain. Values can vary for different registrars. Common values are `registered`, `available`, `CLIENT TRANSFER PROHIBITED`, `CLIENT DELETE PROHIBITED`, `CLIENT UPDATE PROHIBITED`, `clientDeleteProhibited`, `clientTransferProhibited`, `clientUpdateProhibited`.
`registered`	bool	Set if the domain is registered.
`available`	bool	Set if the domain is available.
`created_on`	string	Date and time when the domain record was created. Typically in format YYYY-MM-DD hh:mm:ss TZ, where TZ is timezone information in format +XXXX or timezone abbreviation.
`updated_on`	string	Date and time of the last update of the domain record. Typically

		in format YYYY-MM-DD hh:mm:ss TZ, where TZ is timezone information in format +XXXX or timezone abbreviation.
expires_on	string	Date and time of the domain expiration. Typically in format YYYY-MM-DD hh:mm:ss TZ, where TZ is timezone information in format +XXXX or timezone abbreviation.
registrar	struct	Information about domain registrar. Format is WHOIS_REGISTRAR structure, see below.
registrant_contact	struct	Information about entity that registered the domain. Format is WHOIS_CONTACT structure, see below.
admin_contact	struct	Information about domain administrator. Format is WHOIS_CONTACT structure, see below.
technical_contact	struct	Information about domain technical contact. Format is WHOIS_CONTACT structure, see below.
nameservers	array	Array of WHOIS_NAMESERVER structures, see below.
...		Possibly other values.

WHOIS_REGISTRAR structure is defined as follows (again, not all values are always available):

Name	Type	Description
id	string	Identifier of the registrar.
name	string	Name of the registrar.
organization	string	Name of the registrar's organization.
url	string	Web address of the registrar.

WHOIS_CONTACT structure is defined as follows (again, not all values are always available):

Name	Type	Description
id	string	Name of the domain described by the record.
name	string	Unique domain ID used by some registrars.
organization	string	Contact's organization.
address	string	Contact's address. Usually a street or a full address including a city and a state.
city	string	Contact's city.
zip	string	Contact's ZIP code.
state	string	Contact's state.
country_code	string	Contact's country code.
phone	string	Contact's phone number.
fax	string	Contact's FAX number.
email	string	Contact's email.
created_on	string	Date and time when the contact information was created. Typically in format YYYY-MM-DD hh:mm:ss TZ, where TZ is timezone information in format +XXXX or timezone abbreviation.

WHOIS_NAMESERVER structure is defined as follows (again, not all values are always available):

Name	Type	Description

`name`	string	Host name of the name server.
`ipv4`	string	IPv4 address of the name server.
`ipv6`	string	IPv6 address of the name server.

8.1.2 Failure

Only general errors are returned.

9 Common Structures

`TOOL_STATUS` structure is defined as follows:

Name	Type	Description
`value`	string	Status of the task. Permitted values are: • `OK` – The task finished successfully. • `Partial` – The task finished successfully, but only a part of the result is available. More information is provided in `details`. This status is common when the task is run with a limit and the limit was reached. • `Error` – The task failed. More information is provided in `details`.
`details`	string	Optionally, if `value` is not `OK`, `details` is filled with additional information such as error message or other explanation.

10 Sample Implementation

10.1 PHP 5.3+

```php
<?php
/**
 * Sends an authenticated API call request to the ODT API server.
 *
 * @param string $key
 * @param string $secret
 * @param string $action
 * @param array $postData
 * @return array|mixed
 */

function odtSendRequest($key, $secret, $action, array $postData)
{
  // generate POST data string
  $postFields = http_build_query($postData);
  $dateTime = new \DateTime('now', new \DateTimeZone('UTC'));
  $time = $dateTime->format('Y-m-d H:i:s');

  $message = $key . $time . $postFields;
  $signature = hash_hmac('sha512', $message, $secret);
```

```
  // generate extra headers
  $headers = array(
    'Sign: ' . $signature,
    'Time: ' . $time,
    'Key: ' . $key
  );

  $ch = curl_init();
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($ch, CURLOPT_URL, 'https://secured.online-domain-
tools.com/api/user/' . $action);
  curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
  curl_setopt($ch, CURLOPT_POST, 1);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $postFields);
  curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
  curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
  $res = curl_exec($ch);

  if ($res === false) {
    $result = array(
      'success' => 0,
      'message' => 'Could not get reply: ' . curl_error($ch)
    );
  } else {
    $result = json_decode($res, true);

    if ($result === null) {
      $result = array(
        'success' => 0,
        'message' => 'Invalid response received.'
      );
    }
  }


  return $result;
}
```

11 Changelog

11.1 Version 1.0.0 (24th Aug 2014)

- First public version released.

11.2 Version 1.1.0 (6th Nov 2014)

- Bulk Email Verifier API added.