Alfahosting DNS-API for Resellers

Alfahosting DNS-API for Resellers

This document refers to version 1.0 of the Alfahosting Domain Name System Application Programming Interface for Resellers.

Revision 1.4

Released: June 2, 2014
Author: Oliver Schieche, Alfahosting GmbH
Notes: Updated release

Table of Contents

1. Preamble

1.1. Conventions used throughout the document

The following markup is used in this document:

`identifier`	Text written this way symbolizes an identifier or a symbolic value.

Code snippets look like this:

```
<some code="code" />
```

Hints and remarks to a certain issue are displayed in “hint boxes”:

> **I’m a very important annotation.**

1.2. Disclaimer

Efforts have been taken to ensure completeness and correctness of information given within this document. However, we make no warranties (whether express, implied or statutory) with respect to the information herein.

Alfahosting GmbH cannot be held liable for direct or indirect, consequential or accidental damage resulting from the use of information gathered by this document. At no point in this document shall the reader be granted rights to demand availability or functionality of the product or parts of the product described within this document. We reserve our right to alter or remove certain features of this product without notice.

1.3. Copyright

2. Introduction

2.1. Foreword

This reference manual describes the Application Programming Interface (API) services available to customers and resellers in detail. It is, however, no programming tutorial itself. *The targeted audience is an experienced programmer with profound knowledge of XML and at least one programming language performing calls to this API.*

The manual arranges available commands in the following order:

- User management
- SOA creation / manipulation
- Resource records management

Command descriptions are partitioned into sections labelled “Description”, “Availability”, “Parameters”, “Return Value”, “Validation” and “Error Codes”.

Error codes (see below) resulting from malformed input of any kind are not listed in the overview of possible error codes of a command.

2.2. Alfahosting DNS-System

Our DNS servers are configured to provide real-time domain name resolving.

Changes made to an origin and its corresponding resource records have immediate impact on results of DNS queries.

2.3. Expressions used throughout the document

2.3.1. SOA

Start of Authority – specifies *authoritative* information about a DNS zone, including the primary name server, the email of the domain administrator, the domain serial number, and several timers relating to refreshing the zone.

In this document we informally use the term “zone” to refer to a SOA record.

2.3.2. Refresh

How often a secondary will poll the primary server to see if the serial number for the zone has increased (so it knows to request a new copy of the data for the zone).

2.3.3. Retry

If a secondary was unable to contact the primary at the last refresh, wait the retry value before trying again. It's typically some fraction of the refresh interval.

2.3.4. Expire

How long a secondary will still treat its copy of the zone data as valid if it can't contact the primary. This value should be greater than how long a major outage would typically last, and must be greater than the minimum and retry intervals, to avoid having a secondary expire the data before it gets a chance to get a new copy. After a zone is expired a secondary will still continue to try to contact the primary, but it will no longer provide nameservice for the zone. 2-4 weeks are suggested values.

2.3.5. Minimum

The default TTL (time-to-live) for resource records – how long data will remain in other nameservers' cache. This is by far the most important timer. Set this as large as is comfortable given how often you update your nameserver. If you plan to make major changes, it's a good idea to turn this value down temporarily beforehand. Then wait the previous minimum value, make your changes, verify their correctness, and turn this value back up. 1-5 days are typical values. Remember this value can be overridden on individual resource records.

3. Application Programming Interface

3.1. User relations

The DNS system's management functions can be accessed as a reseller or as a customer. Resellers can create sub-users and have full access to their sub-users' zones and resource records; resellers may be zone-owners as well.

Customers can own and therefore create, edit or delete a zone and its respective resource records.

Resellers can grant or revoke certain features of this product to or from their respective sub-users.

3.2. Terminology

A "request" is constructed within an `<alfadns>` element and contains one or more "calls".

An "element" is an XML-tag with attributes and children (members).

"Calls" are elements with the mandatory attributes `auth`, `uid` and `command`. They can have more mandatory/optional attributes and members.

```
<?xml version="1.0"?>
<element attr="value">
   <member attr="value">CDATA</member>
</element>
```

3.3. Request submission

3.3.1. Input format

All commands to the API are passed via an XML formatted request:

```
<?xml version="1.0"?>
<alfadns login="username">
<call auth="19b288f6…" uid="12345" command="soa.create">
   <soa>mydomain.tld</soa>
   <ns>ns1.namerserver.tld</ns>
   <subuser>sub1dns</subuser>
</call>
</alfadns>
```

Listing 1

The API accepts ISO-8859-1 or UTF-8 input. It will however croak if it encounters mixed input.

3.3.2. Query destination

Submissions are expected to be an HTTP-POST request to the following URLs:

Production System

```
https://dns.alfahosting.de/api/
```

Sandbox (Development)

```
https://dns-test.alfahosting.de/api/
```

Calls to the sandbox are processed the same way calls to the production system are. You can even `dig` our development DNS nameserver to check your expected results:

```
user@localhost:~$ dig @dns-test.alfahosting.de domain.tld

; <<>> DiG 9.3.4-P1.1 <<>> @ dns-test.alfahosting.de domain.tld
; (2 servers found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12345
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 3,
ADDITIONAL: 0

;; QUESTION SECTION:

; domain.tld.                  IN      A


;; ANSWER SECTION:

domain.tld.            86400   IN      A       133.33.33.37


;; AUTHORITY SECTION:

domain.tld.            86400   IN      NS      ns1.nameserver.de.
domain.tld.            86400   IN      NS      ns2.nameserver.de.
domain.tld.            86400   IN      NS      ns3.nameserver.de.

;; Query time: 84 msec
;; SERVER: 133.33.33.37#53(133.33.33.37)
;; WHEN: Tue Jan 06 20:45:00 1982
;; MSG SIZE  rcvd: xxx
```

Changes to the DNS-system's sandbox will be kept for one week.

Access to DNS-API sandbox

In order to access the sandbox you will need to provide a username and sign requests with a key. To retrieve these necessary pieces of information, please contact our support staff via the ticket support system found in your customer's area.

3.3.3. Request response

Responses are given in XML as well:

```
<?xml version=”1.0”?>
<response>
<version>1.0</version>
<call>
   <target>soa.create</target>
   <error>
      <code>-13;EFAILCREATE;an object could not be created</code>
      <details>object:soa;reason:duplicate</details>
   </error>
</call>
</response>
```

Listing 2

Note that each request will be processed in FIFO order and will appear in the response at the according position. A request is encapsulated in a `<call>` element with a child element `<target>` having the value of the request’s command.

3.4. Authorization

All calls have to be authorized in order to be executed. The call attributes `auth` and `uid` are responsible for call authorization. You will have to calculate the HMAC (Hashed Message Authentication Code) Digest for each call. The default behaviour of the API expects the HMAC to be passed as an MD5 32-hex-digit value which is calculated using your API-key and signing the concatenated string

```
command + '-' + uid
```

Listing 3

with it. Where applicable, deviations to this kind of authorization will be provided in a section called "**Authentication Override**." Failure to provide a valid digest results in all calls being deferred.

PHP programmers can rely on the built-in function `hash_hmac` to calculate a fitting HMAC:

```
<?php
   $key = "s3kr3t-k3y";
   $uid = generate_random_string_of_arbitrary_length();
   $command = "soa.create";
   $msg = "${command}-${uid}";

   $hmac = hash_hmac('md5', $msg, $key);
?>
```

Listing 4

> **Note that `hash_hmac` is available to PHP distributions after version 5.1.2.**

3.4.1. Security considerations

In order to provide enhanced security of requests passed to the API it is imperative to use a strong API-key. It should be at least 16 bytes of length and consist of alternations of all characters available from standard ASCII (except 0x00).

For each request, a different `uid` should be used. Consider the following request with a "static" `uid` of `12345` and the key `s3kr3t-k3y`:

```
<?xml version="1.0"?>
<alfadns login="foobert">
   <call command="soa.list"
      uid="12345" auth="3bb35bc2963bbdd16af03467d650c3cd" />
<!-- more calls -->
   <call command="soa.list" subuser="sub2dns"
      uid="12345" auth="3bb35bc2963bbdd16af03467d650c3cd" />
</alfadns>
```

Listing 5

The `uid` attribute can be of arbitrary length and contain any character that won't break XML syntax. It is recommended to use a different `uid` for each call in order to provide a varying HMAC-MD5 digest.

3.5. Result validation

Each API call will yield a corresponding result element containing a `verify` member. This field contains the HMAC-MD5 of the result's verification string whose construction is described in each API call's "**Validation**" section.

Consider a call with the authentication parameters as seen above:

Example

Request:

```
<?xml version="1.0"?>
<alfadns login="foobert">
   <call command="soa.list"
      auth="3bb35bc2963bbdd16af03467d650c3cd" uid="12345">
      <subuser>subfoodns</subuser>
   </call>
</alfadns>
```

Response:

```
<?xml version="1.0"?>
<response>
<version>1.0</version>
<call>
   <result>
      <target>soa.list</target>
      <verify>094b8dc9d45acab77ca40523e566deb0</verify>
      <soa>
         <active>Y</active>
         <expire>604800</expire>
         <ns>ns1.example.com</ns>
         <serial>2009050701</serial>
         <soa>example.com</soa>
         <ttl>86400</ttl>
         <username>subfoodns</username>
      </soa>
      <soa>
         <active>Y</active>
         <expire>604800</expire>
         <ns>ns2.example.com</ns>
         <serial>2009010627</serial>
         <soa>foobert-home.com</soa>
         <ttl>86400</ttl>
         <username>subfoodns</username>
      </soa>
   </result>
</call>
</response>
```

As described in the Validation section of `soa.list`, the `verify` element's value is the response-key signed HMAC of:

```
example.com2009050701foobert-home.com2009010627
```

Validating the response is an obligatory task, yet highly recommended to ensure that the response has been "signed" with your response key.

3.6. User management

The following commands can be used to create, alter or delete sub-users. Note that only resellers have access to the `user.*` commands.

3.6.1. user.list

Description

Enumerate sub-users or retrieve details for a specific sub-user.

Availability

This command is available to resellers with the `user.basic` privilege.

Parameters

The following parameters are used to retrieve a specific sub-user:

Name	Obligation	Type	Description
`subuser`	Optional	A,S	The name of the sub-user to retrieve. The response type (indefinite or specific) depends on this parameter.

These parameters can be used to retrieve a list of sub-users:

Name	Obligation	Type	Description
`offset`	Optional	A,U	An offset from which to begin the result-set with. (Default: 0)
`limit`	Optional	A,U	The number of entries a result-set should not exceed. If set to zero, the result-set contains all sub-users starting at `offset`. (Default: 25)

Return Value (indefinite)

A list of zero or more `<user>` records with the following members will be returned:

Name	Description
`username`	The sub-user's login-name.
`firstname`	The base64-encoded value of the sub-user's first name.
`lastname`	The base64-encoded value of the sub-user's last name.

Return Value (specific query)

Name	Description
`username`	The sub-user's login-name.
`firstname`	The base64-encoded value of the sub-user's first name.
`lastname`	The base64-encoded value of the sub-user's last name.
`street`	Base64 encoded.
`city`	Base64 encoded.
`zip`	
`country`	Base64 encoded.
`max_domains`	
`num_domains`	
`failed_logins`	
`api_privileges`	
`api_access`	A numeric value indicating whether or not the sub-user has access to the API (i.e. user has `api.usage` privilege).

Validation (both)

A string formed like this:

```
fn1:un1:ln1:::fn2:un2:ln2::: … :::fnn:unn:lnn
```

`fn`,`un`,`ln` referring to `firstname`, `username`, `lastname` respectively.

Error Codes

- `ESUBUSERUNSUPPORTED` – when a customer uses this command
- `EOBJECTNOTFOUND` – when a sub-user wasn't found.

3.6.2. user.create

Description

Create a new sub-user.

Availability

This command is available to resellers with the `user.basic` privilege.

Parameters

Name	Obligation	Type	Description	
`type`	Required	ENUM	The subuser type to create. Can be one of "customer" or "reseller"	
`subuser`	Required	S	The login name for the new sub-user.	
`password`	Required	B	A password for front-end usage.	
`max_domains`	Required	U	The maximum number of SOA records the new sub-user is allowed to create.	
`firstname`	Optional	B	Max length 100	Customer info
`lastname`	Optional	B	Max length 100	
`street`	Optional	B	Max length 100	
`city`	Optional	B	Max length 100	
`zip`	Optional	B	Max length 5	
`country`	Optional	B	Max length 100	
`customer_id`	Optional	B	Max length 20	
`api_privileges`	Optional	ACL	The new sub-user's privileges for usage of the API. For valid ACL modifiers, refer to the Privileges section of the Appendix.	API access
`api_key`	Optional	B	The key for signing API requests. Adhere to the guidelines above for maximum strength. Minimum length 12, maximum length of 64 characters.	
`api_response_key`	Optional	B	The key with which an API response is signed. Minimum length 12, maximum length of 64 characters.	

Return Value

A `<success>` element with the new sub-user's login name as value.

Validation

A string formed by concatenating `subuser`, `password` and `max_domains`.

Error Codes

- `ESUBUSERUNSUPPORTED` – when a customer uses this command
- `EFAILCREATE` – the sub-user was not created; this is likely to happen when a sub-user with a given login name already exists. Passwords identical to the login-name, a too short password or API-key or an unknown privilege can also lead to this error. Details are in `<desc>`.
- `ELIMITEXCEEDED` – you have either exceeded the maximum number of sub-users or have attempted to assign a higher `max_domains` value than you have SOA records left. Details are in `<desc>`.

Note that the maximum number of domains for a sub-user is limited by the number of domains already reserved for your existing sub-users and the number of domains granted to you.

3.6.3. user.delete

Description

Delete an existing sub-user and the corresponding resource records and SOA records.

Availability

This command is available to resellers with the `user.basic` privilege.

Parameters

Name	Obligation	Type	Description
`subuser`	Required	A,S	The login name of the sub-user to be deleted.

Return Value

A `<success>` element with the sub-user's login name as value.

Validation

A string with the sub-user's login name.

Error Codes

- `ESUBUSERUNSUPPORTED` – when a customer uses this command.
- `EOBJECTNOTFOUND` – the sub-user could not be found.

3.6.4. user.modify

Description

Change an existing sub-user's properties.

Availability

This command is available to resellers having the `user.basic` privilege.

Parameters

Name	Obligation	Type	Description
`subuser`	Required	A,S	The login name of the sub-user to be edited.
`password`	Optional	S	A new password for the sub-user.
`max_domains`	Optional	U	The new maximum count of domains

			(zones) this sub-user can maintain.	
api_key	Optional	B	The new API-key used to generate API-call HMAC digests with. Make this one at least 16 characters long. Otherwise an error will be reported. Maximum length is 255.	
api_privileges	Optional	ACL	The sub-user's privileges for this API	
failed_logins	Optional	U	Set number of failed logins. Set this to 255 to disable user login (max: 255).	
firstname	Optional	B		Informational
lastname	Optional	B		
street	Optional	B		
city	Optional	B		
zip	Optional	B		
country	Optional	B		

Be advised that at least one of the optional parameters has to be given, lest the API throws an error.

You're also responsible for detecting and avoiding conflicts that could arise from invoking this command. Should you, for example, attempt to set `max_domains` to a value lower than the count of domains (i.e. zones) already maintained by this sub-user, the execution will abort and none of the other properties will be adopted.

Return Value

A `<success>` element with the sub-user's login name as value.

Validation

The input values in the order of the rows of the "Parameters" table concatenated to one string.

Error Codes

- `ESUBUSERUNSUPPORTED` – when a customer uses this command.
- `EOBJECTNOTFOUND` – the sub-user could not be found.
- `EPARAMSYNTAX` – a parameter given failed to validate.

3.7. SOA creation / manipulation

The following commands are available to zone creation and manipulation.

3.7.1. soa.list

Description

Enumerate zones given a specific filter.

Availability

This command is available to users with the `soa.basic` privilege.

Parameters

Name	Obligation	Type	Description
`subuser`	Optional	S+	The zone owners to filter the list by.
`soa`	Optional	A,Z	The zone to find.
`depth`	Optional	ENUM	SELF or ALL
`limit`	Optional	U	Default: 30
`offset`	Optional	U	Default: 0

Return Value

A list of zero or more records with the following members will be returned:

Name	Description
`username`	The sub-user's name who owns this zone.
`soa`	The zone's name.
`ns`	Nameserver for the zone.
`serial`	Serial number of the zone.
`active`	Flag indicating the active state of the zone (either Y or N)

Validation

Each results's `soa` and `serial` field concatenated to one string.

Error Codes

- `ESUBUSERUNSUPPORTED` – when a customer uses the `subuser` parameter

Example

Request:

```
<?xml version="1.0"?>
<alfadns login="username">
  <call auth="…" uid="…" command="soa.list" soa="foobert.com" />
</alfadns>
```

Response:

```
<?xml version="1.0"?>
<response>
   <version>1.0</version>
   <call>
      <target>soa.list</target>
      <verify>1234567890abcdef1234567890abcdef</verify>
      <result>
         <active>Y</active>
         <expire>604800</expire>
         <ns>ns8.alfahosting.info</ns>
         <serial>2009051504</serial>
         <soa>foobert.com</soa>
         <ttl>86400</ttl>
         <username>foobertdns</username>
      </result>
   </call>
</response>
```

3.7.2. soa.create

Description

Create a new Start of Authority record (i.e. zone).

Availability

This command is available to users with the `soa.basic` privilege.

Parameters

Name	Obligation	Type	Description
`subuser`	Optional	S	Customers should skip this parameter; Resellers should pass the name of the sub-user for whom the zone should be created.
`soa`	Required	Z	The new zone's name.
`ns`	Required	D+	The nameservers responsible for the zone (see Remarks).
`ip`	Required	IP	An IPv4 address for the default A-records.
`mbox`	Required	S	The zone's hostmaster's email address.
`refresh`	Optional	U	Refresh interval (def. 28800, min. 3600)
`retry`	Optional	U	Retry interval (def. 7200, min. 3600)
`expire`	Optional	U	Expire interval (def. 604800, min. 86400)
`minimum`	Optional	U	TTL for RR (def. 180, min. 90)
`ttl`	Optional	U	TTL for SOA (def. 86400, min. 7200)
`active`	Optional	ENUM	ENUM(Y|N) Default: Y

All numeric parameters describe time intervals in seconds.

Remarks

Two A-records will be created for this zone pointing to the given `ip` address.

The `ns` parameter is mandatory and must be stated at least once. The first value will be used as the zone's primary nameserver; all additional values will be created as NS-resource records.

Additionally, an MX-resource record with a priority of 10 will be created.

Return Value

An XML-record with the zone's serial as a `serial` member.

Validation

The sum of all numeric parameters (if given; default value otherwise) appended to the first `ns` parameter.

Error Codes

- `EOBJECTNOTFOUND` – sub-user given was not found
- `ELIMITEXCEEDED` – sub-user (or customer, respectively) has reached the limit of SOA records
- `ESUBUSERSUNSUPPORTED` – the parameter `subuser` is required for resellers and required to be omitted by customers
- `EFAILCREATE` – record could not be created; most likely because it already exists.

3.7.3. soa.activate

Description

Toggle a SOA record

Availability

This command is available to users with the `soa.basic` privilege.

Parameters

Name	Obligation	Type	Description
`soa`	Required	A,Z	The zone to change active state for.
`active`	Required	A,ENUM	ENUM(Y|N) The zone's new state.

Return Value

An XML-record with the members `active` and `soa_serial`.

Validation

`soa` and `active` fields concatenated to one string.

Error Codes

- `ESUBUSERUNSUPPORTED` – when a customer uses the `subuser` parameter
- `EOBJECTNOTFOUND` – the zone provided does not exist

Example

Request:

```
<?xml version="1.0"?>
<alfadns login="username">
   <call auth="…" uid="…" command="soa.activate"
      soa="foobert.com" active="N" />
</alfadns>
```

Response:

```
<?xml version="1.0"?>
<response>
   <version>1.0</version>
   <call>
      <target>soa.activate</target>
      <verify>1234567890abcdef1234567890abcdef</verify>
      <result>
         <active>N</active>
      </result>
   </call>
</response>
```

3.7.4. soa.delete

Description

Delete a SOA record and its corresponding resource records.

Availability

This command is available to users having the "`soa.basic`" privilege.

Parameters

Name	Obligation	Type	Description
`soa`	Required	Z+	The zone(s) to delete.

Return Value

A list of zero or more records with the following members will be returned:

Name	Description
`soa`	The zone in question.
`result`	An integer value indicating success or failure: 0: command successful -1: zone not found -2: zone found, but user is not the owner of it
`username`	The name of the sub-user previously associated with the zone. This field is only available for result-code 0.
`rr`	Number of resource records deleted.

Validation

Each result's `result` code and `soa` concatenated into one string.

Authentication Override

Message Digest is calculated as follows:

$$HMAC(command + f_{n-1}() + f_{n-2}() + \ldots + f_0()) \quad ; \text{where } n = count(soa)$$
and
$$f_n() = ':' + uid + soa_n$$

Listing 6

Example

Request:

```
<?xml version=”1.0”?>
<alfadns login=”foobert”>
   <call command="soa.list"
      auth="ec427f8de937ad8e33aec15c50806770" uid="12345">
      <soa>domain1.tld</soa>
      <soa>domain2.tld</soa>
      <soa>domain3.tld</soa>
   </call>
</alfadns>
```

Response:

```
<?xml version=”1.0”?>
<response>
<version>1.0</version>
<call>
   <result>
      <target>soa.list</target>
      <verify>ff605c7513bae5abbdeed09482a2d6de</verify>
      <delete>
         <rr>6</rr>
         <subuser>0wner1dns</subuser>
         <soa>domain1.tld</soa>
         <result>0</result>
      </delete>
      <delete>
         <soa>domain2.tld</soa>
         <result>-1</result>
      </delete>
      <delete>
         <soa>domain3.tld</soa>
         <result>-2</result>
      </delete>
   </result>
</call>
</response>
```

In this example, the request’s authentication code is a result of signing the string

```
soa.delete:12345domain3.tld:12345domain2.tld:12345domain1.tld
```

with our key from the previous examples (*s3kr3t-k3y*).

> **Note that the zones for request authentication have to be given in reverse order.**

3.7.5. soa.modify

Description

Modify a SOA record.

Availability

This command is available to users having the “`soa.basic`” privilege.

Parameters

Name	Obligation	Type	Description
`soa`	Required	A,Z	The name of the zone to be modified.
`ns`	Optional	D	The nameservers responsible for the zone.
`mbox`	Optional	S	The zone’s hostmaster’s email address.
`refresh`	Optional	U	Refresh interval (def. 28800, min. 3600)

`retry`	Optional	U	Retry interval (def. 7200, min. 3600)
`expire`	Optional	U	Expire interval (def. 604800, min. 86400)
`minimum`	Optional	U	TTL for RR (def. 180, min. 90)
`ttl`	Optional	U	TTL for SOA (def. 86400, min. 7200)

Return Value

A record with the following members will be returned:

Name	Description
`soa`	The zone in question.
`modified`	An enumeration of names of properties that have changed. Empty if there were no changes.
`serial`	The new serial for the zone, or the old one if no changes were made.

Validation

The zone's property `soa` and `serial` concatenated into one string.

Error Codes

- `EOBJECTNOTFOUND` – a SOA-record could not be found.

3.8. Resource records management

3.8.1. rr.list

Description

Enumerate resource records given a specific filter.

Availability

This command is available to users with the `rr.basic` privilege.

Parameters

Name	Obligation	Type	Description
`soa`	Required	A,Z	The zone for which resource records shall be enumerated.
`type`	Optional	RR	List resource records of the given type only.

Return Value

A list of zero or more `<rr>` elements with the following members will be returned:

Name	Description
`id`	A unique identifier. Used for deletion via `rr.delete`.
`soa`	The SOA record to which the RR belongs.
`subuser`	The owner of the SOA record of the RR.
`type`	The resource record's type.
`name`	
`data`	
`aux`	
`ttl`	

Validation

Each result's `id`, `type`, `name`, `data`, `aux`, and `ttl` values concatenated to one string.

3.8.2. rr.create

Description
Create a new resource record for a zone.

Availability
The command is available to users having the "`rr.basic`" privilege.

Parameters

Name	Obligation	Type	Description
`soa`	Required	A,Z	The zone to add resource records to.
`type`	Required	RR	Resource record type.
`ttl`	Required	U	The time to live for the record.
`name`	Required	S	See RR specification below.
`data`	Required	S	
`aux`	Required	U	

Return Value
An `<rr>` element with the value 1 as CDATA and the zone's new serial in `soa_serial`.

Validation
None.

Error Codes
- `EOBJECTNOTFOUND` – the `soa` could not be found.
- `EFAILCREATE` – a resource record with the same specs exists.

3.8.3. rr.delete

Description
Delete an RR record from a SOA.

Availability
This command is available to users having the "`rr.basic`" privilege.

Parameters

Name	Obligation	Type	Description
`soa`	Required	A,Z	The zone of which records shall be deleted from.
`rr`	Required	See example	

Return Value
The XML-Element `<rr_deleted>` containing the number of records affected by the deletion command and the affected zone's new serial in `soa_serial`.

Example
Request:

```
<?xml version=”1.0”?>
<alfadns>
   <call auth=”…” uid=”…” command=”rr.delete” soa=”domain.tld”>
      <rr id=”12345” type=”A” />
      <rr id=”23456” type=”A” />
      <rr type=”TXT” />
   </call>
</alfadns>
```

The `<rr>` element’s member `id` can be retrieved via rr.list or can be omitted in which case ***all*** resource records of the given `type` will be deleted.

4. Appendix

4.1. Data types

Parameters whose data type specification have an appended “+” can be stated multiple times.

The following data types are referenced in this document:

Type	Definition
A	Parameter should appear as an attribute of the call.
ACL	Access Control List. See below.
B	A base64 encoded value.
D	A domain name. Do not prepend a protocol (e.g. https://domain.tld)
ENUM	An enumeration of given values. ENUM(Y|N) indicates that a parameter will only accept Y or N as its value.
I	A signed integer.
IP	An IPv4 address.
IP6	An IPv6 address.
RR	Basically an ENUM of all available resource record types: A, AAAA, ALIAS, CNAME, HINFO, MX, NAPTR, NS, PTR, RP, SRV, TXT.
S	A string.
U	An unsigned integer.
Z	A zone name. Actually a plain domain name but without subdomains prepended to it.

Table 1

4.2. Access Control Lists (ACL)

Access Control Lists are constructed by joining multiple access “statements” with a semicolon (‘;’). A statement consists of a modifier and a privilege name. The modifier can be a plus-sign (‘+’) or a minus-sign (‘-‘) to grant or revoke a privilege respectively.

The ACL is parsed in a “first-in, least-effective” manner meaning that statements further to the “right” can override statements given before. Consider the following:

```
+api.usage;+*.basic;-rr.*
```

Listing 7

This would grant a user the `api.usage` privilege, then grant all basic privileges and finally revoke all `rr` privileges.

As seen in Listing 7 wildcards can be used for either the privilege category or the privilege name.

4.2.1. Privileges

The following privileges are available:

Privilege	Description
`api.usage`	General usage of the Application Programming Interface. Without this privilege access will be denied.
`user.basic`	Basic user manipulation commands.
`soa.basic`	Basic SOA record manipulation commands.
`rr.basic`	Basic resource record manipulation commands.

Table 2

4.3. Error Codes

There are two types of errors that can occur: errors indicating overall failure or per-call errors.

4.3.1. General failures

An overall failure presents itself within the `<response>` element an an <error> member, itself with the members `<code>` and `<message>`:

```
<?xml version="1.0"?>
<response>
<version>1.0</version>
<error>
   <code>210</code>
   <message>Too many requests. Your limit is 5 requests.</message>
</error>
</response>
```

Listing 8

Code	Description
-1	All programmers enjoy this type of error. It indicates failure without providing any reason. (Read: unknown error) Any occurrences of this type of error should kindly be reported as bugs.
XML related	
100	An XML-parser error occurred.
101	An XML-syntax or XML-input error occurred. This usually happens when mixed input is encountered by the parser. (Read: non ISO characters and no UTF-8 encoding).
102	The expected request body `alfadns` could not be found.
103	The required `auth` attribute is missing.
104	The required `uid` attribute is missing.
105	CDATA was found between tags. … `<element attr="value">inner text</element>` … In this case, "inner text" is considered to be CDATA. Whitespace does not count as CDATA and will be ignored accordingly.
106	RESERVED
107	Superfluous element found.
108	An empty request was transmitted.
109	Call #xx has no command attribute.
110	RESERVED
Command related	
200	Invalid command.

210	Too many requests.
DNS-API service related	
500	Service unavailable. The user has not been assigned an API-key or lacks the `api.usage` privilege.
510	Authentication parameters are missing.
511	Authentication failed.
520	RESERVED
521	RESERVED

Table 3

4.3.2. Per-call errors

The presence of an `<error>` element within a response's `<call>` element indicates failure. The error element has two or three members: `<code>`, `<details>` and `<desc>` (optional).

The code member is built this way:

```
<code>-1;EPARAMMISSING;a required parameter is missing</code>
```

From this, the actual error code (-1), the error name (EPARAMMISSING) and the human readable reason can be derived.

The following errors can occur:

Code	Name	Description
0	`EINTERNAL`	All programmers enjoy this type of error. It indicates failure without providing any reason.
-1	`EPRIVILEGE`	A privilege requirement for the command was not met.
-10	`EPARAMMISSING`	A required parameter is missing. Details to the missing parameter will be provided.
-11	`EPARAMMISPLACED`	A parameter was expected to appear as a call's attribute but was found as a call's member (also vice versa; details are provided).
-12	`EPARAMSYNTAX`	A parameter's value is malformed. Most likely a result of invalid values passed to the parameter.
-13	`EPARAMSINGLETON`	A parameter was expected to appear only once but was found more than once.
-14	`EPARAMUNKNOWN`	A parameter unknown to the command was given.
-15	`EPARAMMISMATCH`	Provided parameters clash. Usually happens, when you query for a single value and provide an offset greater than zero.
-20	`EOBJECTNOTFOUND`	An object could not be found. Most likely results from passing a non-existing sub-user or zone.
-21	`ESUBUSERUNSUPPORTED`	A customer called a function and provided a `subuser` parameter. See `User relations`.
-22	`ELIMITEXCEEDED`	A hard-limit has been reached.
-23	`EFAILCREATE`	An object could not be created.
-24	`EEXECABORT`	A command could not be successfully

		executed. Details can be found in `desc`.

Table 4

4.4. Resource Record Specification

For the `rr.create` command the API will perform certain sanity checks for the parameters provided. Although all parameters are listed as “required”, with the exception of the `ttl` parameter, they aren’t. Refer to the following sections to find out which resource records expect certain parameters to be present.

4.4.1. A record

This is an “address record.” It holds an IPv4 address, most commonly used to map hostnames to an IP address of the host. Requires the following parameters:

- `name` – the hostname (use ‘*’ or ‘*.subdomain’ for wildcard entries)
- `data` – an IPv4 address

4.4.2. AAAA record

This is also an address record. It holds a 128-bit IPv6 address. Required parameters are:

- `name` – the hostname (use ‘*’ or ‘*.subdomain’ for wildcard entries)
- `data` – an IPv6 address

4.4.3. ALIAS record

This is a server-side alias, treated almost like a CNAME, only it’s handled entirely by the server. Expects the following parameters:

- `name` – the alias’ name
- `data` – the host for which the alias stands

The hostname for which the alias should resolve to has to be another SOA in our Domain Name System.

ALIAS records will appear as A records to the client.

4.4.4. CNAME record

A canonical name record is an alias of one name to another: the DNS lookup will continue by retrying the lookup with the new name. Requires:

- `name` – canonical name for the alias
- `data` – the real name of the machine specified by `name`. This can be a hostname (“`foo`”) or an FQDN (“`foo.domain.tld.`”).

4.4.5. HINFO record

Informational record providing host information. The `data` member should contain two strings specifying CPU type (1st string) and operating system type (2nd string). If either string needs to contain a space, enclose it in quotation marks. Well-formed values can be found in RFC 1700 (“Assigned Numbers”) on pages 214 (“OPERATING SYSTEM NAMES”) and 206 (“MACHINE NAMES”) respectively.

4.4.6. MX record

A Mail Exchange record. Maps a domain name to a list of mail exchange servers for that domain. Wants the following parameters:

- `data` – a hostname or an FQDN which will accept mail for the host specified by `name`.
- `name` – the host for which mail should be accepted.
- `aux` – a preference value for the mail server.

> **Mail transfer agents prefer MX records with the lowest preference value first. This value *can* be zero.**

4.4.7. NAPTR record

The naming authority pointer record's `data` parameter should contain a base64 encoded string consisting of six white-space separated fields which describe a regular expression based rewrite rule as described in RFC 2915 ("The Naming Authority Pointer (NAPTR) DNS Resource Record").

4.4.8. NS record

An authoritative nameserver record. Expects the following parameters:

- `data` – a hostname or an FQDN of a server considered to be an authoritative nameserver for the zone specified by `name`.
- `name`

4.4.9. PTR record

A domain name pointer. These records, used only with IN-ADDR.ARPA zones should contain the canonical hostname of the machine referred to by `name` in `data`.

4.4.10. RP record

A responsible person. Has nothing in common with M.A.D.D.
Refer to RFC 1183 ("New DNS RR Definitions"), section 2.2 for details.
Applicable content should be stored in the `data` parameter and base64 encoded.

4.4.11. SRV record

Generalized service location record, used for newer protocols instead of creating protocol-specific records such as MX. These parameters are expected:

- `name` – contains service and protocol specification and an optional sub-domain name. The format of this value is an all lowercase case string stating both service and protocol name as strings prefixed with an underscore, followed by a sub-domain name and all concatenated with a single dot. While the service name is an arbitrary value (only restricted to alphanumerical characters), the protocol *must* be either "_tcp", "_udp" or "_tls". An SRV record defining an HTTP server on the sub-domain "foo" would thus have a `<name>` value of "_http._tcp.foo" (excluding quotes).

- `data` – must contain three space-separated values specifying the weight for this entry, the number of the port on the target host of the service and a hostname or FQDN of the target host. Again, RFC 1700 (“Assigned Numbers”) can be referred to for well-known services (page 15, “WELL KNOWN PORT NUMBERS”).
- `aux` – the priority of the entry. As with MX records, lower priority entries are preferred.

Further information may be gathered through RFC 2782 (“A DNS RR for specifying the location of services (DNS SRV)”).

4.4.12. TXT record

A text string. Content is held in the `data` field and needs to be base64 encoded.

4.5. Changelog

- 2009-05-15 – initial version
- 2010-05-26 – fixed a typo in code sample for HMAC
- 2010-06-03 – added `api_response_key` parameter to user.create
- 2014-05-26 – added missing SRV record documentation for the record's service and protocol
  (thanks for pointing that out, unnamed customer :-))
- 2014-06-02 – added missing details regarding hostname in SRV records

<<EOF