RcodeZero DNS

Documentation & Workflow

REST-API

Document category: Operative Documentation	Author: ipcom GmbH Creation/Change Date: 20200907	Version: V2 Status: Final
Classification: public	T +43 662 46 69 –0 Mail: rcodezero@ipcom.at	Effective from: 20201002 Effective till: -
Distribution list:	-	
	Prüfung: PR, 20201002	Freigabe: PR, 20201002

Content

Document history (optional):

Date	Version	Name	Changes
23.9.2020	V.1.	Marketing & PR, Michael Braunöder	Creation of document, Upload pdf. On website 29.9. Changes Wording in general Instead of “Master” -> Primary Instead of “Slave” -> Secondary 4.2. Rate Limits: Adjustment Rate Limit: `X-RateLimit-Limit` from 200 to 1000
02.10.2020	V.2.	Michael Braunöder	4.2. Rate Limits: Adjustment Text: From: Authenticated requests are limited to 1000 requests per minute and IP address -> To: Authenticated requests are limited to 1000 requests per minute and account

1 OPENAPI 3.0 SPECIFICATION

This API is documented in the OpenAPI 3.0 format. The OpenAPI 3.0 Json file is available at https://my.rcodezero.at/openapi/rcode0api-latest.json.
Changelog and a rendered version of the API is available https://my.rcodezero.at/openapi/.

2 OVERVIEW

RcodeZero DNS is a globally distributed, easy to use Anycast DNS network. The RcodeZero Anycast network is provisioned via web interface or REST API and supports

- Primary Zones: no infrastructure on customer side is needed, configuration of zone entirely via web interface or API.
- Secondary Zones: requires a hidden primary name server on customer side. The RcodeZero network receives DNS NOTIFYs on changes, and subsequently performs transfers via Zone transfer (initial zone add is performed via API or web)

All zones can be signed with DNSSEC.

3 INFRASTRUCTURE

3.1 CLOUDS

The RcodeZero Anycast Network provides DNS service via two independend clouds. For resilience, each cloud has its own IP range (IPv4 and IPv6), a different set of locations and a seperate routing policy.
Each newly added zone is assigned to an instance on each cloud and the cloud ip addresses are returned to the customer.

3.2 CONTROL SERVERS

The Control Server are the “gateways” to the RcodeZero DNS Network. The control server consumes DNS Notifies from customers, transfers zone from hidden primaries and redistributes zone in the anycast network.
The needed information which RcodeZero IP ranges must be allowed for zone transfers, where the notfifies have to be sent to and from which IP address the zones can be transferred out can be found in the RcodeZero Dashboard (https://my.rcodezero.at/login).

3.3 TEST SYSTEM

RcodeZero DNS provides a functionally identical test system. This is intended to in order to 'play' with the system or the API, and allows new customers to try out RcodeZero DNS. The test system is available at https://my-test.rcodezero.at.

The limitations of the test system with respect to the production system are:

- No Service Level Agreements. While we strive to keep the test system available all the time, there may be outages due to deployment of new software revisions or other operational reasons.
- Only 2 unicast name server nodes (the production system has 20+ anycast nodes).
- May use newer software to beta-test new features.

To obtain an account for the test system please sent us an email to rcodezero@ipcom.at.

4 REST-API

4.1 AUTHENTICATION

This API uses Bearer Token for authentication. Obtain your Bearer token via the web interface https://my.rcodezero.at/enableapi. This token must be included as an HTTP Header in each API request.

```
Authorization: Bearer <token>
```

Unauthorized request will be rejected with HTTP error code 401.

4.2 RATE LIMITS

- IP addresses (IPv4, for IPv6 the corresponding /64) originating unauthorized requests will be blocked for 10 minutes after 10 requests. Further requests will be rejected with the HTTP Error code 429.
- Authenticated requests are limited to 1000 requests per minute and account. Request over the limit will be rejected with the HTTP error code 429.
- The Rate Limit and the remaining requests are reported in every response with the headers.

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 199
```

If a request has been rejected an additional header contains the timestamp when the limit will be reseted:

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1523441309
```

4.3 IP FILTER

Accounts can be limited to defined Source IP addresses or subnet. This can be configured via the web interface. Requests from other IP addresses are rejected with HTTP error code 401.

4.4 PAGINATION

Request that might return a lot of data like GET /zones or GET /reports/problematiczones are paginated. To get the next page use the ?page parameter. The default number of items per page is 100 and can be changed via the ?page_size parameter.

4.5 RESPONSES

The REST call return HTTP Status codes 2xx for successfull calls, and 4xx/5xx if an error occured.
If a call does not return objects (e.g. if a action is processed), the call return a structure with the attributes “status” and “message”. If the action has been processed without an error the response looks like.

```
{
  "status": "ok",
  "message": "Zone testzone1.at successfully added"
}
```

Otherwise a “status: failed” with an error message will be returned.

```
{
  "status": "failed",
  "message": <error message>
}
```

5 WORKFLOWS

5.1 PRIMARY ZONES

- Add zone via API or web
- Configure RRSets via API or web (the serial number of the SOA-record is updated automatically)
- Changes are distributed automatically to the RcodeZero Anycast Network. Changes can take up to two minutes until the become visible in the DNS.

5.2 SECONDARY ZONES

This is a suggested workflow to activate RcodeZero Anycast DNS for a secondary zone. Of course this work flow can be adopted to fit your needs.

- Configure your primary name server(s) to allow zone transfers towards our control name server (IP addresses listed below). If the primary name server is a “hidden primary”, also allow our control name server to query the primary(s) (for serial number checks).
- Add the zone via the API or web
- Add the RcodeZero name server(s) to the NS RRSet of your zone. Either use the RcodeZero host names mentioned in the dashboard or use your own branded name servers. For branded nameservers, make sure their A/AAAA records point to the IP addresses of the returned nameservers.

Example with RcodeZero hostnames:

```
example.com.  IN  NS sec1.rcode0.net.
example.com.  IN  NS sec2.rcode0.net.
```

Example with RcodeZero hostnames:

```
example.com.  IN  NS ns1.provider-xyz.net.
example.com.  IN  NS ns2.provider-xyz.net.

ns1.provider-xyz.net. IN A    192.174.68.100
ns1.provider-xyz.net. IN AAAA 2001:67c:1bc::100
ns2.provider-xyz.net. IN A    176.97.158.100
ns2.provider-xyz.net. IN AAAA 2001:67c:10b8::100
```

NOTE: If your contract includes dedicated IP addresses or hosting your own network, obviously you should use the respective addresses.

Instead of adding RcodeZero Secondary DNS as additional name server by adding a new NS record to the zones, you can also replace one of your existing name servers with RcodeZero. In this case, you only have to change the A/AAAA records of the existing name server hostname to point to our anycast IP addresses. This does not require any changes to zones and registry – as long as the hostname is outside the zone and thus requires no glue records.

The primary name server must send DNS NOTIFYs in order to trigger zone transfers. If NOTIFYs are not sent, the control name server checks for zone updates every “refresh” seconds, according to the value of the “refresh” field in the zone’s SOA record. To reduce SOA-query load on the control name server, a refresh value of 24h is used if the original value is lower than 24h. In all cases, it is mandatory for the primary name server to increase the serial on zone changes. Otherwise, the changes will be ignored by RcodeZero. Additionally, zone transfers can be requested via the “retrieve” command on the API interface. In this case, the serial is ignored and the zone is transferred and deployed to the Anycast nodes even if the serial has not increased.

NOTE: Even after a successful zone transfer, the Anycast name servers may respond with stale data for a few minutes due to caching in the name server software

5.3 DNSSEC

RcodeZero supports pre-signed (secondary) zones (the zone is signed on the hidden primary on customer side) or signs (if wanted) the zone before distributing it on the RcodeZero network. Both primary and secondary zones can be signed with the RcodeZero DNSSEC service.
The DNSSEC workflow contains some asynchronous processes. These processes generate notifies when they have finished and other transactions can be performed. The following notifies are used:

- DSSEEN: We see the active DS-Record in the parent zone and finish the initial signing or key rollover process. A new KSK rollover can now be performed.
- DSUPDATE: A new KSK has been pre-published and it is save now to publish the DS record in the parent zone
- DSREMOVED: The DS record of the active KSK has been removed from the parent zone and the TTLs and the propagation delay has been expired. It is save now to perform an unsign transaction.

The notifies are added to the message queue of your account and can be retrieved and acknowledged via API or web interface. If your account has a notification E-mail address configured (via web interface) an additional email is sent to this address.

So, what is the typical RcodeZero DNSSEC workflow?

- add a domain: POST /zone
- sign the domain: POST /zone/<zone>/sign
- poll the message queue until the DSUPDATE event for the zone is retrieved: GET /messages
- ack the message: DELETE /messages/<id>
- get the zone details to grab the DNSKEY/DS record: GET /zone/<zone>
- Add the DNSKEY/DS to the parent zone
- Acknowledge that the DNSKEY/DS has been added to the parent zone: POST /zone//dsupdate

5.3.1 Important DNSSEC aspects

- **Serial**

DNSSEC requires periodical resigning of the zone, thus the zone serial number needs to be increased. Therefore, for signed zones the zone's serial announced by the Anycast nodes will be greater than the serial number on the customers hidden primary server. But, a higher serial on the Anycast node is not an indication that the zone is up2date. Therefore, every time the serial is increased on the hidden primary, the new serial number should be greater than the serial on the anycast node.

A convenient solution is to use the Unix epoch at the time of the zone update as the zone serial numer. This eases debugging and ensures proper zone updates even after un-signing a zone.

- **Hidden Primary**

For zones using the DNSSEC signing service, the customers primary name server must be a hidden primary and the zone must not be delegated to that hidden primary. Further, if the zone is also hosted on a customers name server, this name server must not transfer the zone from the (customer) hidden primary, but must transfer the signed zone from the RcodeZero Anycast control server (see "set secondaries" below). This is also required for parent zones.
Consider the following problematic example:

The name server ns1.isp.com hosts the zone example.com:

```
zone example.com:
aexample.com.         NS ns1.isp.com.
signed.example.com. NS sec1.rcode.net.
```

Further, this name server acts as hidden primary for the zone signed.example.com. The zone is unsigned delivered to RcodeZero, and signed by RcodeZero. Therefore, the zone's name server only point to the RcodeZero Anycast name servers:

```
zone signed.example.com:
www.signed.example.com. A  1.1.1.1
signed.example.com.     NS sec1.rcode.net.
```

The problem here is, that resolvers which trace www.signed.example.com from the root zone on its way also query ns1.isp.com. Although ns1.isp.com is a hidden primary for signed.example.com, it will authoritatively answer the request for www.signed.example.com without DNSSEC signatures. Therefore the validation will fail.
The hidden primary should not be queried by public resolvers – even for other zones.

5.4 ALIAS (ANAME) SUPPORT

ALIAS/ANAME is a feature to allow CNAME-like forwarding also on zone apex. ALIAS is the proprietary name introduced by the PowerDNS developers, ANAME will the new name once the feature becomes an RFC. See https://tools.ietf.org/html/draft-ietf-dnsop-aname.
RcodeZero supports the ALIAS record using the proprietary record type 65401 as implemented within PowerDNS. When RcodeZero receives an A or AAAA request for a label which does have an ALIAS record, RcodeZero resolves the ALIAS target and responds to the request with the resolved ALIAS target.
For Primary DNS it is quite simple to use: just select ALIAS as record type and provide the ALIAS target (a hostname, no IP address) as FQDN with trailing dot.
For Secondary DNS just provision the ALIAS record on your primary name server. How this is done exactly depends on your name server software. When using PowerDNS >= 4.0 you can just insert an ALIAS type into the records table. For other name server software you can specify the ALIAS in a generic, encoded form. For example: example.com should be an ALIAS to www.example.com:
PDNS >=4.0:
example.com. IN ALIAS www.example.com.
PDNS <4.0, Bind9:
example.com. IN TYPE65401 # 17 03777777076578616d706c6503636f6d00

Note: Using ALIAS with Secondary DNS is not supported on DNSSEC signed zone - regardless if you sign yourself or use the DNSSEC signing service of RcodeZero! ALIAS with Secondary DNS will work only on unsigned zones.
As ALIAS support is currently quite low, we provide you with an online converter:
https://my.rcodezero.at/alias.php?alias=www.example.com

Here is the sample code of the converter to integrate it into your systems:

```
<?php

  $alias = "www.exmaple.com.";

  $alias = trim($alias," \t\n\r\0\x0B.");
  if (preg_match("/[^A-Za-z0-9-_.]/", $alias)) {
    die("ERROR: Invalid Characters. Allowed: A-Za-z0-9-_.\n");
  }


  $labels = explode(".",$alias);
  $enc="";

  foreach ($labels as $label) {
    if (strlen($label) > 63) {
      die("ERROR: label $label is longer than 63 characters!\n");
    }

    // encode the length
    $enc .= sprintf('%02x', strlen($label));

    // encode the label
    foreach (str_split($label) as $char) {
      $enc .= sprintf('%02x', ord($char));
    }
  }

  // add trailing 00
  $enc .= sprintf('%02x', 0);

  echo "IN A TYPE65401 \# " . strlen($enc)/2 . " " . $enc . "\n";

?>
```