API V2.0

Documentation

7/28/2014

Table of Contents

Revision History

21-May-12... Initial Release

07-June-12 ....Data Types & API Calls Update

25-Sept-13.. Added Multi Create/Delete

28-July-14................................ Error Correction, Added Domain/Record Search

Overview

Making A Request

REST requests can be made via HTTPS or HTTP by using the headers, authentication, data types, and methods specified below. The current endpoint for the API V2.0 is available at:

http[s]://api.dnsmadeeasy.com/V2.0/

The Sandbox

The sandbox is available as a non-production environment to test API calls and any code developed to run on top of the API. A sandbox account is completely independent from a production account in DNS Made Easy and has different API keys.

A sandbox account can be created at:

http://sandbox.dnsmadeeasy.com/account/new

The current endpoint for the sandbox API is available at:

http[s]://api.sandbox.dnsmadeeasy.com/V2.0/

Rate Limiting

To prevent unwanted flooding of the API system, there is a maximum number of requests that can be sent in a given time period. This limit is 150 requests per 5 minute scrolling window. For example, 100 requests could be made in one minute, followed by a 5 minute wait, following by 150 requests. This limit is tracked per API key and all requests count toward this limit. Refer to the *x-dnsme-requestLimit* and *x-dnsme-requestsRemaining* header fields for values related to this limit.

Supported Data Formats

The DNS Made Easy API supports both XML and JSON data formats, specified by using the *content-type* and *accept* HTTP header fields. If no format is specified, JSON will be used as the default.

Authentication

Authentication with the DNS Made Easy API is done using the API and Secret keys, given on a per-account basis. The values for these keys can be found on the **Config - Account Information** page once logged into your DNS Made Easy account. To make an authenticated request, follow these steps:

- Create the string representation of the current UTC date and time in HTTP format. Example: `Sat, 12 Feb 2011 20:59:04 GMT`
- Calculate the hexadecimal HMAC SHA1 hash of that string using your Secret key as the hash key. Example: `b3502e6116a324f3cf4a8ed693d78bcee8d8fe3c`
- Set the values for the request headers using your API key, the current date and time, and the HMAC hash that you calculated. This example was created using a Secret key of c9b5625f-9834-4ff8-baba-4ed5f32cae55:
  - `x-dnsme-apiKey:1c1a3c91-4770-4ce7-96f4-54c0eb0e457a`
  - `x-dnsme-requestDate:Sat, 12 Feb 2011 20:59:04 GMT`
  - `x-dnsme-hmac:b3502e6116a324f3cf4a8ed693d78bcee8d8fe3c`

Requests must be sent shortly after these headers are generated. If too much time has passed between when the *x-dnsme-requestDate* and *x-dnsme-hmac* strings were created and when the request is received by the DNS Made Easy API servers, then the request will be denied.

Requests made with invalid credentials or an invalid *x-dnsme-requestDate* value will receive an **HTTP 403 – Forbidden response**.

Common Header Fields

The DNS Made Easy API includes several custom HTTP header fields that contain information about the requests and responses that are sent. These headers fields are:

Request Header Fields

Field name	**Description**
x-dnsme-apiKey	The API Key for your account. Refer to the **Config – Account Information** menu once logged in to find this value.
x-dnsme-requestDate	Standard HTTP-formatted date. This date is used to protect against falsified requests played back at a future time.
x-dnsme-hmac	HMAC hash of value of the x-dnsme-requestDate field. Refer to the DNS Made Easy REST API Documentation v2.0 Authentication section for more details on how to generate this value.

Response Header Fields

Field name	**Description**
x-dnsme-requestId	A unique identifier of the API call that was sent. Use the value of this header to help identify your request for your own purposes or when contacting DNS Made Easy support.
x-dnsme-requestLimit	Maximum number of requests that can be sent before the rate limit is exceeded.
x-dnsmerequestsRemaining	Number of requests remaining before the rate limit is exceeded.

Try it! Getting Started

If you'd like to make an API request, just follow these steps. We've provided a sample Perl script that can be used to make requests.

- Download and install cURL. Make sure that the cURL executable is part of your path.
- Download and install Perl. Make sure that the Perl executable is part of your path.
- Download the necessary Perl extensions by running the following commands:

  ```
  perl -MCPAN -e "install Digest::HMAC_SHA1"
  perl -MCPAN -e "install HTTP::Date"
  perl -MCPAN -e "install Config::Properties"
  ```

- Download the following DNS Made Easy files and save them into the same location:
  - dnsmeapi.properties
  - dnsmeapi.pl
- Put your API and Secret Keys into the dnsmeapi.properties file, the location of these keys is specified below in the Authentication section.
- Make requests using the dnsmeapi.pl script as a wrapper around cURL! For example:

```
perl dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/ -H
accept:application/xml

perl dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/12345 -X

perl dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/
dns/managed/12345/records -H accept:application/json
```

cURL accepts the -X option to set the HTTP method used for the request (GET is used if no method is specified). cURL accepts the –H option to set HTTP headers for the request.

API Calls

The following is a detailed listing of all data types used for requests and responses with the DNS Made Easy API. Not seeing a specific example API call you are looking for? Please contact sales@dnsmadeeasy.com to request it be added.

Managed DNS Fields - /dns/managed/

The dns/managed/ data type contains information about a single domain. The following fields are available:

/dns/managed/

Field Name	Type	Description
name	String	The domain name
id	numeric	The domain ID
nameServers	List of Strings	Name servers assigned to the domain by DNS Made Easy *(System defined, unless vanity is applied)*
gtdEnabled	boolean	Indicator of whether or not this domain uses the Global Traffic Director service
soaID	numeric	The ID of a custom SOA record
templateId	numeric	The ID of a template applied to the domain
vanityId	numeric	The ID of a vanity DNS configuration
transferAclId	numeric	The ID of an applied transfer ACL
folderId	numeric	The ID of a domain folder
updated	numeric	The number of seconds since the domain was last updated in Epoch time
created	numeric	The number of seconds since the domain was last created in Epoch time
axfrServer	List of Strings	The list of servers defined in an applied AXFR ACL.
delegateNameServers	List of Strings	The name servers assigned to the domain at the registrar

Example XML Representation

```
<domain>
```

```
        <created>2014-02-19T00:00:00+00:00</created>
        <delegateNameServers>ns0.dnsmadeeasy.com.</delegateNameServers>
        <delegateNameServers>ns1.dnsmadeeasy.com.</delegateNameServers>
        <delegateNameServers>ns2.dnsmadeeasy.com.</delegateNameServers>
        <delegateNameServers>ns3.dnsmadeeasy.com.</delegateNameServers>
        <delegateNameServers>ns4.dnsmadeeasy.com.</delegateNameServers>
        <folderId>12345</folderId>
        <gtdEnabled>false</gtdEnabled>
        <id>1234567</id>
        <name>example.com</name>
        <nameServers>
                <fqdn>ns0.dnsmadeeasy.com</fqdn>
                <ipv4>208.94.148.2</ipv4>
                <ipv6>2600:1800:0::1</ipv6>
        </nameServers>
        <nameServers>
                <fqdn>ns1.dnsmadeeasy.com</fqdn>
                <ipv4>208.80.124.2</ipv4>
                <ipv6>2600:1801:1::1</ipv6>
        </nameServers>
        <nameServers>
                <fqdn>ns2.dnsmadeeasy.com</fqdn>
                <ipv4>208.80.126.2</ipv4>
                <ipv6>2600:1802:2::1</ipv6>
        </nameServers>
        <nameServers>
                <fqdn>ns3.dnsmadeeasy.com</fqdn>
                <ipv4>208.80.125.2</ipv4>
                <ipv6>2600:1801:3::1</ipv6>
        </nameServers>
        <nameServers>
                <fqdn>ns4.dnsmadeeasy.com</fqdn>
                <ipv4>208.80.127.2</ipv4>
                <ipv6>2600:1802:4::1</ipv6>
        </nameServers>
        <templateId>1234</templateId>
        <updated>2014-07-10T18:02:30.051+00:00</updated>
</domain>
```

Example JSON Representation

```
{
"name":"myDomain.com",
"nameServer":[
"ns10.dnsmadeeasy.com",
"ns11.dnsmadeeasy.com",
"ns12.dnsmadeeasy.com",
"ns13.dnsmadeeasy.com",
"ns14.dnsmadeeasy.com",
"ns15.dnsmadeeasy.com"
],
"gtdEnabled":true}

{
      "name":"example.com",
      "id":1234567,
      "folderId":12345,
      "nameServers":[
            {
                  "fqdn":"ns0.dnsmadeeasy.com",
                  "ipv6":"2600:1800:0::1",
                  "ipv4":"208.94.148.2"
            },
            {
                  "fqdn":"ns1.dnsmadeeasy.com",
                  "ipv6":"2600:1801:1::1",
                  "ipv4":"208.80.124.2"
            },
            {
                  "fqdn":"ns2.dnsmadeeasy.com",
                  "ipv6":"2600:1802:2::1",
                  "ipv4":"208.80.126.2"
            },
            {
                  "fqdn":"ns3.dnsmadeeasy.com",
                  "ipv6":"2600:1801:3::1",
                  "ipv4":"208.80.125.2"
            },
            {
                  "fqdn":"ns4.dnsmadeeasy.com",
                  "ipv6":"2600:1802:4::1",
                  "ipv4":"208.80.127.2"
            }],
      "gtdEnabled":false,
      "updated":1405015350051,
      "templateId":1234,
      "delegateNameServers":[
            "ns0.dnsmadeeasy.com.",
            "ns1.dnsmadeeasy.com.",
            "ns2.dnsmadeeasy.com.",
           "ns3.dnsmadeeasy.com.",
           "ns4.dnsmadeeasy.com"],
      "created":1392768000000
}
```

SINGLE DOMAIN ACTIONS

The following calls are actions for a single domain name.

Searching for specific Domains by ID or name

To return information about a single domain, you must first have its associated domain ID.

To view a full list of all domains and domain ID's in your account issue the call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/
```

You can also find a specific domain ID by domain name:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/managed/name?domainn
ame={domainname}


./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/managed/id/{domainna
me}
```

Managed DNS GET- Return a domain

The following call would return configuration information of a single domain including name server assignment and the domain ID.

Example Call – Return data about a single domain with the domain ID 999999:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/999999
```

Example Output:

```
{"name":"example.com","id":999999,"gtdEnabled":true,"nameServers":[{"i
pv4":"208.94.148.4","fqdn":"ns10.dnsmadeeasy.com","ipv6":"2600:1800:10
::1"},{"ipv4":"208.80.124.4","fqdn":"ns11.dnsmadeeasy.com","ipv6":"260
0:1801:11::1"},{"ipv4":"208.80.126.4","fqdn":"ns12.dnsmadeeasy.com","i
pv6":"2600:1802:12::1"},{"ipv4":"208.80.125.4","fqdn":"ns13.dnsmadeeas
y.com","ipv6":"2600:1801:13::1"},{"ipv4":"208.80.127.4","fqdn":"ns14.d
nsmadeeasy.com","ipv6":"2600:1802:14::1"},{"ipv4":"208.94.149.4","fqdn
":"ns15.dnsmadeeasy.com","ipv6":"2600:1800:15::1"}],"pendingActionId":
```

```
0,"folderId":99999,"created":1326758400000,"delegateNameServers":[],"u
pdated":1337121058848}
```

Example Error: Generic API 404 Error for a badly formulated request.

Managed DNS PUT- Update a domain

This call would be used to change a configuration option of a single domain such as assigned vanity or template.

Example Call – Changing the Vanity DNS ID to 9999 for domain ID 999999:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/999999 -X
PUT -H accept:application/json -H content-type:application/json -d
'{"vanityId":"9999"}'
```

Example Error:

```
{"error":["Invalid domain IDs specified.","Invalid Vanity NS
configuration."]}
```

Either an invalid domain ID or Vanity DNS ID was specified.

Managed DNS POST- Create a domain

This call would be used to create a single domain.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/ -X POST -H
accept:application/json -H content-type:application/json -d
'{"name":"example.com"}'
```

Example Output:

```
{"name":"example.com","id":99999,"folderId":1171,"gtdEnabled":false,"u
pdated":1406046257363,"nameServers":[{"fqdn":"ns0.dnsmadeeasy.com","ip
v4":"208.94.148.2","ipv6":"2600:1800:0::1"},{"fqdn":"ns1.dnsmadeeasy.c
om","ipv4":"208.80.124.2","ipv6":"2600:1801:1::1"},{"fqdn":"ns2.dnsmad
eeasy.com","ipv4":"208.80.126.2","ipv6":"2600:1802:2::1"},{"fqdn":"ns3
.dnsmadeeasy.com","ipv4":"208.80.125.2","ipv6":"2600:1801:3::1"},{"fqd
n":"ns4.dnsmadeeasy.com","ipv4":"208.80.127.2","ipv6":"2600:1802:4::1"
}],"pendingActionId":1,"processMulti":false,"activeThirdParties":[],"c
reated":1405987200000}
```

Example Error:

```
{"error":["Domain name conflicts with existing zones."]}
```

The domain name already exists within a DNS Made Easy account.

Managed DNS DELETE- Delete a domain

This call would be used to delete a single domain.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/999999 -X
DELETE -H accept:application/json -H content-type:application/json -d
'{["999999"]}'
```

Example Error:

```
{"error":["Cannot delete a domain that is pending a create or delete
action."]}
```

The domain is already in creating or deleting status and cannot be deleted until this pending action completes.

MULTIDOMAIN ACTIONS

The following calls would be used to perform a single action on many domains at once, such as creation or deletion. In addition to the available fields from */dns/managed* data type, the following fields are also available for multi-domain actions:

/dns/managed

Field Name	**Type**	**Description**
names	List of Strings	List of domain names
ids	List of numbers	List of domain identifiers

Managed DNS GET- Return all domains

This call will return all the managed DNS domains in the account including the domain name, domain ID, and additional GTD, folder, and date configured information.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/
```

Example Output:

```
{"data":[{"name":"example.com","id":999999,"gtdEnabled":false,"pending
ActionId":0,"folderId":99999,"created":1336953600000,"updated":1337033
603667},{"name":"example1.com","id":999991,"gtdEnabled":true,"pendingA
ctionId":0,"folderId":99999,"created":1326758400000,"updated":13371210
58848},{"name":"example2.com","id":999992,"gtdEnabled":false,"pendingA
ctionId":0,"folderId":9999,"created":1331164800000,"updated":133122508
7653},{"name":"example3.com","id":999993,"gtdEnabled":false,"pendingAc
tionId":0,"folderId":9999,"created":1334016000000,"updated":1334080916
510}],"page":1,"totalPages":1,"totalRecords":6}
```

Example Error:

If the account has no domains, an empty request is returned: { }

Managed DNS PUT- Update multiple domains

This call would be used to change configuration of multiple domains.

Example Call – Changing the Vanity DNS ID to 9999 for domain ID’s 999999 and 999991:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed -X PUT -H
accept:application/json -H content-type:application/json -d
'{"ids":["999999","999991"],"vanityId":"9999"}'
```

Example Error:

```
{"error":["Invalid domain IDs specified.","Invalid Vanity NS
configuration."]}
```

Either an invalid domain ID or Vanity DNS ID was specified.

Managed DNS POST- Create multiple domains

This call would be used to create multiple domains.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed -X POST -H
accept:application/json -H content-type:application/json -d
'{"names":["example01.com","example02.com"]}'
```

Example Output:

```
[999991,999992]
```

The domain ID's for the newly created domains.

Example Error:

```
{"error":["Domain name conflicts with existing zones."]}
```

One of the domain names exists within a DNS Made Easy account already.

Managed DNS DELETE – Delete multiple domains

This call would be used to delete multiple domains with ID's 999999 and 999991.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed -X DELETE -
H accept:application/json -H content-type:application/json -d
'[999999,999991]'
```

Example Output: No output on successful call

Example Error:

```
{"error":["Cannot delete a domain that is pending a create or delete
action."]}
```

Record Fields - /dns/managed/{domain_id}/records/

The record data type contains information about DNS records for a given domain. The following fields are available:

/dns/managed/{domain_id} /records/		
Field Name	**Type**	**Description**
name	string	The record name
value	string	HTTPRED: <redirection URL> MX: <priority> <target name> NS: <name server> PTR: <target name> SRV: <priority> <weight> <port> <target name> TXT: <text value>
id	numeric	The unique record identifier
type	string	The record type. Values: A, AAAA, ANAME, CNAME, HTTPRED, MX, NS, PTR, SRV, TXT, SPF, or SOA.
source	numeric	1 if the record is the record is domain specific, 0 if the record is part of a template.
sourceID	numeric	The source domain ID of the record.
dynamicDns	boolean	Indicates if the record has dynamic DNS enabled or not.
password	String	The per record password for a dynamic DNS update.
ttl	numeric	The time to live or TTL of the record.
monitor	boolean	Indicates if System Monitoring is enabled for an A record.
failover	boolean	Indicates if DNS Failover is enabled for an A record.
failed	boolean	Indicates if an A record is in failed status.
gtdLocation	string	Global Traffic Director location. Values: DEFAULT, US_EAST, US_WEST, EUROPE
password	string	For A records that have dynamic DNS. Password used to authenticate for dynamic DNS.
description	string	For HTTPRED records. A description of

		the HTTPRED record.
keywords	string	For HTTPRED records. Keywords associated with the HTTPRED record.
title	string	For HTTPRED records. The title of the HTTPRED record.
redirectType	string	For HTTPRED records. Type of redirection performed. Values: Hidden Frame Masked, Standard – 302, Standard – 301

hardlink	Boolean	For HTTPRED records
mxLevel	numeric	The priority for an MX record
weight	numeric	The weight for an SRV record
priority	numeric	The priority for an SRV record
port	numeric	The port for an SRV record

Example XML Representation

```
<record>
      <dynamicDns>false</dynamicDns>
      <failed>false</failed>
      <failover>false</failover>
      <gtdLocation>DEFAULT</gtdLocation>
      <hardLink>false</hardLink>
      <id>12345678</id>
      <monitor>false</monitor>
      <name>test</name>
      <source>1</source>
      <sourceId>1234567</sourceId>
      <ttl>1800</ttl>
      <type>A</type>
      <value>1.1.1.1</value>
</record>
<record>
      <dynamicDns>false</dynamicDns>
      <failed>false</failed>
      <failover>false</failover>
      <gtdLocation>DEFAULT</gtdLocation>
      <hardLink>false</hardLink>
      <id>12345679</id>
```

```
	<monitor>false</monitor>
	<mxLevel>5</mxLevel>
	<name>test</name>
	<source>1</source>
	<sourceId>1234567</sourceId>
	<ttl>1800</ttl>
	<type>MX</type>
	<value>mx1.mailserver.com.</value>
</record>
```

Example JSON Representations

```
{
	"name":"test",
	"value":"1.1.1.1",
	"id":12345678,
	"type":"A",
	"source":1,
	"gtdLocation":"DEFAULT",
	"failed":false,
	"failover":false,
	"sourceId":1234567,
	"monitor":false,
	"dynamicDns":false,
	"ttl":1800,
	"hardLink":false
}
{
	"name":"test",
	"value":"mx1.mailserver.com.",
	"id":12345679,
	"type":"MX",
	"source":1,
	"gtdLocation":"DEFAULT",
	"failed":false,
	"failover":false,
	"sourceId":1234567,
	"monitor":false,
	"dynamicDns":false,
	"ttl":1800,
	"mxLevel":5,
	"hardLink":false
}
```

Searching for specific Records by name or type

To perform operations on a specific record in a domain, you must first obtain the associated record ID for that record.

To view a full list of all records within a specific domain you would issue the call:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records
```

To search for specific record(s) by type and/or name:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/managed/{domainId}/r
ecords?type={recordType}

./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/managed/{domainId}/r
ecords?recordName={recordName}&type={recordType}
```

Managed DNS Records GET- Return records

This call would display all the records in a single domain ID and returns the record ID's of each associated record.

Example Call – View records in domain ID 999999:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records
```

Example Output:

```
{"data":[{"name":"mail","value":"www","id":9999999,"type":"CNAME","ttl
":1800,"sourceId":999999,"failover":false,"monitor":false,"gtdLocation
":"DEFAULT","source":1,"dynamicDns":false,"failed":false,"hardLink":fa
lse},{"name":"google","value":"http://www.google.com/","id":9999991,"t
ype":"HTTPRED","ttl":1800,"sourceId":999999,"failover":false,"monitor"
:false,"title":"title","gtdLocation":"DEFAULT","redirectType":"Standar
d -
302","keywords":"keywords","description":"description","source":1,"dyn
amicDns":false,"failed":false,"hardLink":false},{"name":"www","value":
"5.5.5.5","id":9999992,"type":"A","ttl":1800,"sourceId":999999,"failov
er":false,"monitor":false,"gtdLocation":"EUROPE","source":1,"dynamicDn
s":false,"failed":false,"hardLink":false}],"page":1,"totalPages":1,"to
talRecords":3}
```

Example Error: Generic API 404 Error for a badly formulated request.

Managed DNS Record PUT- Update a record

This call will update a record with ID 2222222 in the domain with ID 999999. The call would change the A record mail to the IP 1.1.1.1 with a TTL of 86400 seconds.

Example Call:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records/2222222/ -
X PUT -H accept:application/json -H content-type:application/json -d
'{"name":"mail","type":"A","value":"1.1.1.1","id":"2222222","gtdLocati
on":"DEFAULT","ttl":86400}'
```

Example Output: No output on successful call.

Example Error: Generic API 404 Error for a badly formulated request.

Managed DNS Record POST- Create a record

This call is used to create a record in a domain and will return the recordID.

Example Call – Create a record in the domain with domain ID 999999:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records/ -X POST -
H accept:application/json -H content-type:application/json -d
'{"name":"test","type":"A","value":"1.1.1.1","gtdLocation":"DEFAULT","
ttl":86400}'
```

Example Output:

```
{"name":"test","value":"1.1.1.1","id":9999993,"type":"A","ttl":86400,"
sourceId":999999,"failover":false,"monitor":false,"gtdLocation":"DEFAU
LT","source":1,"dynamicDns":false,"failed":false,"hardLink":false}
```

Example Error:

```
{"error":["Record with this type (A), name (test), and value (1.1.1.1)
already exists."]}
```

The A record with the name and IP entered already exists.

Managed DNS Record DELETE – Delete a record

Delete a single DNS record a domain.

Example Call – Delete the record with recordID 1234567 from the domain with domainID 999999:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records/1234567

-X DELETE
```

Example Output: No output on successful call.

Example Error: Generic API 404 Error for a badly formulated request.

Multi Record Fields - /dns/managed/{domain_id}/multi/records/

The multi record data type contains information about domain actions for multiple records at a time. The following fields are available:

/dns/managed/{domain_id}/multi/records/

Field Name	**Type**	**Description**
name	string	The record name
value	string	HTTPRED: <redirection URL> MX: <priority> <target name> NS: <name server> PTR: <target name> SRV: <priority> <weight> <port> <target name> TXT: <text value>
id	numeric	The unique record identifier
type	string	The record type. Values: A, AAAA, ANAME, CNAME, HTTPRED, MX, NS, PTR, SRV, TXT, SPF, or SOA.
source	numeric	1 if the record is the record is domain

		specific, 0 if the record is part of a template.
sourceID	numeric	The source domain ID of the record.
dynamicDns	boolean	Indicates if the record has dynamic DNS enabled or not.
password	string	The per record password for a dynamic DNS update.
ttl	numeric	The time to live or TTL of the record.
monitor	boolean	Indicates if System Monitoring is enabled for an A record.
failover	boolean	Indicates if DNS Failover is enabled for an A record.
failed	boolean	Indicates if an A record is in failed status.
gtdLocation	string	Global Traffic Director location. Values: DEFAULT, US_EAST, US_WEST, EUROPE
password	string	For A records that have dynamic DNS. Password used to authenticate for dynamic DNS.
description	string	For HTTPRED records. A description of the HTTPRED record.
keywords	string	For HTTPRED records. Keywords associated with the HTTPRED record.
title	string	For HTTPRED records. The title of the HTTPRED record.
redirectType	string	For HTTPRED records. Type of redirection performed. Values: Hidden Frame Masked, Standard – 302, Standard – 301
hardlink	boolean	For HTTPRED records
mxLevel	numeric	The priority for an MX record
weight	numeric	The weight for an SRV record
priority	numeric	The priority for an SRV record
port	numeric	The port for an SRV record

Multi-Record POST- Create multiple records in a single domain

This call will create multiple records under a single domain in a single call.

Example Call:

```
./dnsmeapi.pl
http://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records/createMulti
-X POST -H accept:application/json -H content-type:application/json -d
'[{"name":"test", "type":"CNAME", "value":"google.com.",
"gtdLocation":"DEFAULT","ttl":1800}, {"name":"test1", "type":"CNAME",
"value":"google.com.", "gtdLocation":"DEFAULT","ttl":1800},
{"name":"test2", "type":"CNAME", "value":"google.com.",
"gtdLocation":"DEFAULT","ttl":1800}]'
```

Example Output:

```
[{"name":"test","value":"google.com.","id":12345678,"type":"CNAME","so
urce":1,"gtdLocation":"DEFAULT","failed":false,"failover":false,"sourc
eId":1234567,"monitor":false,"dynamicDns":false,"ttl":1800,"hardLink":
false},{"name":"test1","value":"google.com.","id":12345679,"type":"CNA
ME","source":1,"gtdLocation":"DEFAULT","failed":false,"failover":false
,"sourceId":1234567,"monitor":false,"dynamicDns":false,"ttl":1800,"har
dLink":false},{"name":"test2","value":"google.com.","id":12345670,"typ
e":"CNAME","source":1,"gtdLocation":"DEFAULT","failed":false,"failover
":false,"sourceId":1234567,"monitor":false,"dynamicDns":false,"ttl":18
00,"hardLink":false}]
```

```
The records with associated ID’s.
```

Example Error:

```
{"error":["Duplicate name and value match detected for record \"test1
google.com.\" with domain \"example.com\"."
```

A record already exists with this name in this domain, conflicting with creating a CNAME of the same name.

Multi-Record PUT – Update multiple records in a single domain

This call will update multiple records for a single domain.

Example Call:

```
./dnsmeapi.pl
http://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records/updateMulti
-X PUT -H accept:application/json -H content-type:application/json -d
'[{"id":9938153, "name":"test", "type":"CNAME",
"value":"www.google.com.", "gtdLocation":"DEFAULT","ttl":1800},
{"id":9938155, "name":"test1", "type":"CNAME",
"value":"www.google.com.", "gtdLocation":"DEFAULT","ttl":1800},
{"id":9938154, "name":"test2", "type":"CNAME",
"value":"www.google.com.", "gtdLocation":"DEFAULT","ttl":1800},
{"id":9938154, "name":"test3", "type":"CNAME",
"value":"www.google.com.", "gtdLocation":"DEFAULT","ttl":1800}]'
```

Example Error:

```
{"error":["Duplicate name and value match detected for record \"test1
google.com.\" with domain \"example.com\"."
```

Multi-Record DELETE – Delete multiple records from a single domain

This call is used to delete multiple records from a domain.

Example Calls:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records?ids=999999
9, 9999991, 9999992, 9999993 -X DELETE
```

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/managed/999999/records?ids=999999
9&ids=23123123&ids=891293 -X DELETE
```

Example Output: No output on successful call.

SOA Record Fields - /dns/soa/

The SOA Record data type contains parameters for custom SOA records and their configuration. The following fields are available.

/dns/soa

Field Name	Type	Description

name	string	SOA Record name
id	numeric	Identifier (system defined)
email	string	Contact email address
ttl	numeric	TTL of SOA record (in seconds)
comp	string	Primary name server
serial	numeric	Starting zone serial number
refresh	numeric	Zone refresh time (in seconds)
retry	numeric	Failed Refresh retry time (in seconds)
expire	numeric	Expire time of zone (in seconds)
negativeCache	numeric	Record not found cache (in seconds)

Example XML Representation

```
<data type="soa">
      <comp>ns10.dnsmadeeasy.com.</comp>
      <email>dns.dnsmadeeasy.com.</email>
      <expire>604800</expire>
      <id>1234</id>
      <name>Master For Second Set</name>
      <negativeCache>10800</negativeCache>
      <refresh>14400</refresh>
      <retry>3600</retry>
      <serial>2009010102</serial>
      <ttl>21600</ttl>
</data>
```

Example JSON Representation

```
{
      "name":"Custom SOA",
      "id":1234,
      "email":"dns.dnsmadeeasy.com.",
      "comp":"ns10.dnsmadeeasy.com.",
      "refresh":14400,
      "serial":2009010102,
      "retry":3600,
      "expire":604800,
      "negativeCache":10800,
      "ttl":21600
}
```

SOA Record GET- Return SOA records

Example Call - This call will display all custom SOA records defined for an account:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/soa/ -H
accept:application/json -H content-type:application/json -X GET
```

Example Output:

```
{"data":[{"name":"Master For Second
Set","id":1234,"email":"dns.dnsmadeeasy.com.","comp":"ns10.dnsmadeeasy
.com.","refresh":14400,"serial":2009010102,"retry":3600,"expire":60480
0,"negativeCache":10800,"ttl":21600},{"name":"Master for First
Set","id":1235,"email":"dns.dnsmadeeasy.com.","comp":"ns0.dnsmadeeasy.
com.","refresh":43200,"serial":2008010102,"retry":3600,"expire":120960
0,"negativeCache":180,"ttl":86400}],"page":0,"totalPages":1,"totalReco
rds":2}
```

Example Error: If no records exist an empty set is returned.

Example Call – Return a Single SOA with ID 1234

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/soa/1234
```

SOA Record PUT – Update an SOA record

Example Call – Assigning a custom SOA to domains. This call will assign the domains with ID's 999999 and 999991 with the custom SOA record with ID 9999:

```
 ./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/ -H
accept:application/json -H content-type:application/json -X PUT -d
'{"ids":["999999","999991"], "soaId":"9999"}'
```

Example Call – Editing a specific SOA with SOA ID 1234

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/soa/1234 -X PUT -d
'{"name":"My SOA
Record","id":1234,"email":"dns.dnsmadeeasy.com.","comp":"ns0.dnsmadeea
sy.com.","refresh":43200,"serial":2008010102,"retry":3600,"expire":120
9600,"negativeCache":180,"ttl":86402}'
```

SOA Record POST – Create an SOA record

Example Call - This call will create the domains example.com and example1.com and assigned the custom SOA with ID 9999:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/managed/ -H
accept:application/json -H content-type:application/json -X POST -d
'{"names":["example.com","example1.com"], "soaId":"9999" }'
```

SOA Record DELETE – Delete an SOA record

Example Call – This call will delete an SOA record with ID 1234 (must not be applied to any domains at the time of deletion):

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/soa/1234 -X DELETE
```

Vanity DNS Fields - /dns/vanity/

The Vanity DNS data type contains parameters for vanity DNS configuration.

/dns/vanity/		
Field Name	**Type**	**Description**
name	string	The identifiable name of the Vanity DNS group
id	numeric	Identifier (system defined)
nameServerGroupId	numeric	The name server group the config is assigned
publicConfig	boolean	True represents a system defined rather than user defined vanity configuration.
defaultConfig	boolean	Indicates if the vanity configuration is the system default
servers	string	The vanity host names defined in the config
Default	boolean	True sets the configuration as the default
nameServerGroup	string	Lists the DNS Made Easy name servers the configuration is defined for.

Example XML Representation

```
<data type="vanity">
      <default>false</default>
      <id>123456</id>
      <name>Custom Vanity Configuration</name>
      <nameServerGroup>ns0,ns1,ns2,ns3,ns4.dnsmadeeasy.com</nameServerG
      roup><nameServerGroupId>1</nameServerGroupId>
      <public>false</public>
      <servers>ns2.example.com</servers>
      <servers>ns3.example.com</servers>
</data>
```

Example JSON Representation

```
{
      "name":"Custom Vanity Configuration",
      "id":123456,
      "nameServerGroupId":1,
      "nameServerGroup":"ns0,ns1,ns2,ns3,ns4.dnsmadeeasy.com",
      "servers":["ns2.example.com","ns3.example.com"],
      "public":false,
      "default":false
}
```

Vanity Name Server GET- Return all vanity name server configurations

This call will display a full list of all vanity name server groups public and private defined within an account.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/vanity
```

Example Output:

```
{"data":[{"name":"Example
Vanity","id":9999,"nameServerGroupId":1,"public":false,"servers":["ns0
.example.com","ns1.example.com"],"default":false,"nameServerGroup":"ns
0,ns1,ns2,ns3,ns4.dnsmadeeasy.com"}
],"totalRecords":1,"totalPages":1,"page":1}
```

Vanity Name Server PUT- Edit an existing vanity name server configuration

Example Call – Update an existing Vanity Name Server configuration with ID 1234:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/vanity/1234 -X PUT
-H accept:application/json -H content-type:application/json -d
'{"name":"Update
Name","id":1234,"nameServerGroupId":1,"nameServerGroup":"ns0,ns1,ns2,n
s3,ns4.dnsmadeeasy.com","servers":["ns0.example.com","ns1.example.com"
],"public":false,"default":false}'
```

Example Output: No output on successful call.

Vanity Name Server POST- Create a new vanity name server configuration

Example Call – Create a new vanity configuration:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/vanity/ -X POST -H
accept:application/json -H content-type:application/json -d
'{"name":"New
Vanity","nameServerGroupId":1,"nameServerGroup":"ns0,ns1,ns2,ns3,ns4.d
nsmadeeasy.com","servers":["ns2.example.com","ns3.example.com"],"publi
c":false,"default":false}'
```

Example Output:

```
{"name":"New
Vanity","id":12345,"nameServerGroupId":1,"nameServerGroup":"ns0,ns1,ns
2,ns3,ns4.dnsmadeeasy.com","servers":["ns2.example.com","ns3.example.c
om"],"public":false,"default":false}
```

The new vanity configuration and associated ID.

Vanity Name Server DELETE – Delete a vanity DNS configuration

Example Call – This call will delete a vanity DNS configuration with ID 1234 (must not be applied to any domains at the time of deletion):

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/vanity/1234 -X
DELETE
```

Template Fields- /dns/template/

The Template data type contains parameters for Record Set Template configuration.

/dns/template/

Field Name	**Type**	**Description**
name	string	Template name
id	numeric	Identifier *(System Defined)*
domainIds	numeric	Domain ID's currently assigned to the template
publicTemplate	boolean	True represents a system defined public template rather than user defined account specific template.

Example XML Representation

```
<data type="template">
	<id>12345</id>
	<name>Custom Template</name>
	<publicTemplate>false</publicTemplate>
</data>
```

Example JSON Representation

```
{
	"name":"Custom Template",
	"id":12345,
	"domainIds":[],
	"publicTemplate":false
}
```

TEMPLATE ACTIONS

Template GET- Return template configurations

Example Call - This call would return all templates public and private within the account including:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/template/
```

Example Output:

```
{"name":"Example
Template","id":9999,"domainIds":[],"publicTemplate":false}
```

A full list of templates and associated ID's.

Example Call – This call will return a single template by ID:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/template/12345
```

Example Error: Generic API 404 Error for a badly formulated request.

Template POST- Create a new template configuration

Example Call- This call will create a new public template:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/template/ -X POST -d
'{"name":"New Template","publicTemplate":false}'
```

Example Output:

```
{"name":"New
Template","id":14685,"domainIds":[],"publicTemplate":false}
```

The template data and associated ID.

Template DELETE – Delete a template configuration

Example Call – This call will delete a template with ID 1234:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/template/1234 -X
DELETE
```

Example Output – No output on successful call.

TEMPLATE RECORD ACTIONS

Template Record GET- Return template records

Example Call - This call would return all records within template ID 9999

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/
dns/template/9999/records/
```

Example Call - This call would return all A records within template ID 9999:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/
dns/template/9999/records?type=A
```

Example Output:

```
{"data":[{"name":"www","value":"1.1.1.1","id":9999999,"type":"A","sour
ce":2,"gtdLocation":"DEFAULT","ttl":1800,"sourceId":9999,"failover":fa
lse,"monitor":false,"hardLink":false,"dynamicDns":false,"failed":false
}],"page":1,"totalPages":1,"totalRecords":1}
```

Example Error: Generic API 404 Error for a badly formulated request.

Template Record PUT- Update template records

Example Call: This call updates a record with record ID 12345678 in template ID 123456

```
./dnsmeapi.pl
http://api.dnsmadeeasy.com/V2.0/dns/template/123456/records/12345678/
-X PUT -d
'{"name":"","value":"mail","id":12345678,"type":"MX","ttl":1800,"mxLev
el":10}'
```

Example Output – No output on successful call.

Template Record POST- Create template records

This call will create an A record within the template with the ID 9999.

Example Call:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/template/9999/records/ -X POST -H
accept:application/json -H content-type:application/json -d
'{"name":"www","type":"A","value"2.2.2.2","gtdLocation":"DEFAULT","ttl
":86400}'
```

Example Output:

```
{"name":"www","value":"2.2.2.2","id":9999999,"type":"A","ttl":86400,"s
ourceId":9999,"failover":false,"monitor":false,"gtdLocation":"DEFAULT"
,"source":2,"dynamicDns":false,"failed":false,"hardLink":false}
```

Example Error: Generic API 404 Error for a badly formulated request.

Template Record DELETE- Delete template records

Example Call - This call will delete the record with ID 9999999 from the template with ID 9999:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/template/9999/records?ids=9999999
-X DELETE
```

Account ACL Fields - /dns/transferAcl/

The Account ACL data type contains parameters for Access Control Lists defined for the account.

/dns/transferAcl/

Field Name	Type	Description
name	string	ACL Identifiable name
id	numeric	Identifier (system defined)
ips	numeric	The IP addresses defined in the ACL

Example XML Representation

```
<data type="transferAcl">
	<id>12345</id>
	<ips>1.1.1,1</ips>
	<name>Custom ACL</name>
</data>
```

Example JSON Representation

```
{
	"name":"Custom ACL",
	"id":12345,
	"ips":["1.1.1.1"]
}
```

Transfer ACL GET- Return ACL configurations

Example Call- This call will display a full list of all AXFR transfer ACL's:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/transferAcl
```

Example Output:

```
{"data":[{"name":"TestACL","id":9999,"ips":["1.2.3.4"]}],"totalRecords
":1,"totalPages":1,"page":1}
```

Example Call – This call will display a single transfer ACL with ID 1234:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/transferAcl/1234/
```

Transfer ACL PUT- Update an ACL

Example Call – This call will update an existing transfer ACL with ID 1234, changing the name and adding a new IP:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/transferAcl/1234/ -X
PUT -d '{"name":"Transfer ACL
Update","id":1234,"ips":["1.1.1.1",”2.2.2.2”]}'
```

Transfer ACL POST- Create a new ACL

Example Call – This call will create a new Transfer ACL:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/transferAcl/ -X POST
-d '{"name":"Test","ips":["3.3.3.3"]}'
```

Example Output:

```
{"name":"Test","id":1234,"ips":["3.3.3.3"]}
```

The ACL data with associated ID.

Transfer ACL DELETE – Delete an ACL

Example Call – This call will delete a transfer ACL with ID 1234:

```
./dnsmeapi.pl http://api.dnsmadeeasy.com/V2.0/dns/transferAcl/1234 -X
DELETE
```

Folder Fields - /security/folder

The Folder data type contains parameters for managing domain folders defined in the account.

/security/folder/		
Field Name	**Type**	**Description**
value	numeric	Identifier (system defined)
label	string	The name of the folder
/security/folder/<Folder ID>		
name	string	The name of the folder
id	numeric	The ID of the folder
Domains	List of strings	A list of the primary domain ID's assigned to the folder
secondaries	List of strings	A list of the secondary domain ID's assigned to the folder
folderPermissions	List of strings	A list of the permissions for the folder
defaultFolder	boolean	Indicator of the folder being marked as Default.

Folder GET- Return folder information

Example Call - This call will display a full list of all folders defined in the account.

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/security/folder
```

Example Output:

```
[{"value":99999,"label":"Default"},{"value":99991,"label":"Folder1"}]
```

A list of all folders and associated ID's.

Example Call- This call will display information about a single folder including domains contained within it by ID:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/security/folder/12345
```

Example Output:

```
{"name":"Folder1","id":12345,"domains":[12345678,12345679],"secondarie
s":[],"folderPermissions":[{"permission":0,"folderId":12345,"groupId":
11111,"folderName":"Folder1","groupName":"Group1"},{"permission":0,"fo
lderId":12345,"groupId":22222,"folderName":"Folder1","groupName":"Defa
ult"}],"defaultFolder":true}
```

Folder PUT – Update a Folder

Example Call – This call will update a folder changing its name:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/security/folder/12345 -
X PUT -d '{"name":"New Folder
Update","id":12345,"defaultFolder":true}'
```

Example Output: No output on successful call.

Folder POST – Create a new Folder

Example Call – The following call creates a new folder adding the domain with ID 1234567 to the folder:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/security/folder/ -X
POST -d '{"name":"New Folder
Create","domains":[1234567],"defaultFolder":false}'
```

Example Output:

```
{"name":"New Folder
Create","id":93710,"domains":[1442922],"secondaries":[],"folderPermiss
ions":[{"folderId":93710,"folderName":"New Folder
Create"}],"defaultFolder":false}
```

The created folder with associated ID.

Folder DELETE – Remove a configured Folder

Example Call – Delete a Folder with ID 12345:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/security/folder/12345 -
X DELETE
```

Example Output – No output on successful call.

Query Usage Fields - /usageApi/queriesApi/

The query usage data type contains information for account query usage by month, day, domain, or per month per domain.

/usageApi/queriesApi/		
Field Name	**Type**	**Description**
id	numeric	identifier
total	numeric	Query total for the specified day/month/year
month	numeric	Month of the year
day	numeric	Day of the month
year	numeric	Year
accountId	numeric	Account identifier
primaryCount	numeric	Count of primary domains
primaryTotal	numeric	Query count total for primary domains
secondaryCount	numeric	Count of secondary domains
secondaryTotal	numeric	Query count total for secondary domains

Example XML Representation:

```
<queryUsage>
      <total>17372</total>
      <month>12</month>
      <year>2011<year>
      <accountId>99999</accountId>
</queryUsage>
```

Example JSON Representation:

```
[{
      "id":null,
      "total":1712,
      "month":9,
      "year":2011,
      "day":null,
      "accountId":99999,
      "primaryCount":0,
      "primaryTotal":0,
      "secondaryCount":0,
      "secondaryTotal":0
},
{
      "id":null,
      "total":2386,
      "month":10,
      "year":2011,
      "day":null,
      "accountId":99999,
      "primaryCount":0,
      "primaryTotal":0,
      "secondaryCount":0,
      "secondaryTotal":0
}]
```

Query Usage GET – Display all Query Usage

This call will display a full report of query traffic within the account for all months and all domains.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/usageApi/queriesApi
```

Display Query Usage for a single month

This call will display query usage for April of 2012 for all domains.

Example Call:

```
./dnsmeapi.pl  https://
api.dnsmadeeasy.com/V2.0/usageApi/queriesApi/2012/4
```

Query Usage for a Single Month for a Single Domain

This call will display query usage for April of 2012 for the domain with ID 999999.

Example Call:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/usageApi/queriesApi/2012/4/managed/99999
9
```

Failover Fields - /monitor/

The Failover data type contains parameters for DNS Failover and System Monitoring configuration.

/monitor/		
Field Name	**Type**	**Description**
monitor	boolean	True indicates System Monitoring Enabled
recordId	numeric	The record of ID is the record with failover configured
systemDescription	string	The system description configured for the failover event notification.
maxEmails	numeric	The number of emails sent to the contact for an outage.
sensitivity	numeric	The number of checks placed against the primary IP before a Failover event occurs. List of Sensitivity ID's: Low (slower failover) = 8 Medium = 5 High = 3

protocolId	numeric	The protocol for DNS Failover to monitor on.
		List of Protocol IDs:
		TCP = 1
		UDP = 2
		HTTP = 3
		DNS = 4
		SMTP = 5
		HTTPS = 6

port	numeric	The port for DNS Failover to monitor on the specified protocol.
failover	boolean	True indicates DNS Failover Enabled
autoFailover	boolean	True indicates the failback to the primary IP address is a manual process. False indicates

		the failback to the primary IP is an automatic process.
ip1	numeric	The primary IP address
ip2	numeric	The secondary IP address
ip3	numeric	The tertiary IP address
ip4	numeric	The quaternary IP address
ip5	numeric	The quinary IP address
Ip1Failed	numeric	Indicates if IP is currently in failed status and how many times it has failed.
Ip2Failed	numeric	Indicates if IP is currently in failed status and how many times it has failed.
Ip3Failed	numeric	Indicates if IP is currently in failed status and how many times it has failed.
Ip4Failed	numeric	Indicates if IP is currently in failed status and how many times it has failed.
Ip5Failed	numeric	Indicates if IP is currently in failed status and how many times it has failed.
source	numeric	1 indicates the record is part of the domain, 0 indicates the record is part of a template
sourceId	numeric	The source ID of the domain or template for the record.
contactListId	numeric	The ID of the contact list for system monitoring notifications
httpFqdn	string	The FQDN to monitor for HTTP or HTTPS checks.
httpFile	string	The file to query for for HTTP or HTTPS checks.
httpQueryString	string	The string to query for for HTTP or HTTPS checks.

DNS Failover GET- Return a DNS Failover configuration for a record

This call will display a DNS Failover for a record with ID 1234567.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/monitor/1234567 -X GET
```

Example Output:

```
{"port":80,"source":1,"failover":true,"ip1":"1.1.1.1","ip2":"2.2.2.2",
"protocolId":3,"sourceId":1046205,"monitor":true,"sensitivity":5,"syst
emDescription":"Test","maxEmails":1,"ip1Failed":0,"ip2Failed":0,"ip3Fa
iled":0,"ip4Failed":0,"ip5Failed":0,"recordId":1234567,"autoFailover":
false}
```

DNS Failover PUT- Update a DNS Failover configuration for a record

This call will update (or create initially) a DNS Failover configuration for a record with ID 1234567.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/monitor/1234567 -X PUT
-d
'{"port":8080,"failover":true,"ip1":"1.1.1.1","ip2":"2.2.2.2","protoco
lId":3,"monitor":true,"sensitivity":5,"systemDescription":"Test","maxE
mails":1,"autoFailover":false}'
```

DNS Failover PUT - Disable DNS Failover for a Record

Example Call – This call will disable DNS Failover for a record with ID 1234567:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/monitor/1234567 -X PUT
-d '{"port":81,"failover":false,"monitor":false,"sensitivity":5}'
```

Secondary DNS Fields - /dns/secondary

Secondary DNS GET – Return secondary DNS domains

Example Call - This call will return all secondary DNS domains configured in the account:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary
```

Example Call – This call will return the secondary DNS domain with ID 1234567:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary/1234567
```

Example Output:

```
{"name":"example.com","id":1234567,"folderId":12345,"nameServers":[{"f
qdn":"ns5.dnsmadeeasy.com","ipv6":"2600:1800:5::1","ipv4":"208.94.148.
13"},{"fqdn":"ns6.dnsmadeeasy.com","ipv6":"2600:1801:6::1","ipv4":"208
.80.124.13"},{"fqdn":"ns7.dnsmadeeasy.com","ipv6":"2600:1802:7::1","ip
v4":"208.80.126.13"}],"nameServerGroupId":100,"pendingActionId":0,"gtd
Enabled":false,"updated":1395422608376,"ipSet":{"name":"IP Set
12345","id":12345,"ips":["2.2.2.2"]},"ipSetId":12345,"created":1395360
000000}
```

Secondary DNS PUT – Change the IP Set of a secondary domain

This call will update the secondary domains with ID's 123456 and 123457 assigning the IP Set with ID 99999.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary -X PUT -H
accept:application/json -H content-type:application/json -d
'{"ids":["123456","123457"],"ipSetId":"99999"}'
```

Example Output: No output on successful call.

Example Call – Change the IP Set of a single secondary domain with ID 123456:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary/123456 -X
PUT -H accept:application/json -H content-type:application/json -d
'{"ipSetId":"12345","folderId":11111}'
```

Secondary DNS POST - Create secondary DNS domains

This call will create domains under secondary DNS management with an assigned IP Set. An IP Set must be assigned at creation.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary -X POST -
H accept:application/json -H content-type:application/json -d
'{"names":["example.com","example1.com"],"ipSetId":"99999"}'
```

Example Output: The created domain ID's

```
[99999,99991]
```

Secondary DNS DELETE – Delete secondary DNS domains

This call will delete a secondary domain with ID 123456.

Example Call:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary/123456 -X
DELETE
```

Example Output: No output on successful call.

IPSet Fields - /dns/ipSet

IP Set GET – Return a list of IP Sets

Example Call - This call will return a list of secondary IP Sets:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary/ipSet -X
GET
```

Example Output:

```
{"data":[{"name":"Master
Set","id":12345,"ips":["2.2.2.2"]},{"name":"Second Master
Set","id":12346,"ips":["1.1.1.1"]}],"page":1,"totalPages":1,"totalReco
rds":2}
```

Example Call – Return a single IP Set with ID 12345:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/secondary/ipSet/12345
```

IP Set PUT - Change the name or IP's in an IP Set

Example Call – Update IP Set 12345 to edit the name and IP:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/secondary/ipSet/12345/ -X PUT -d
'{"name":"IP Set Update","id":12345,"ips":["2.2.2.1"]}'
```

IP Set POST – Create a new IP Set

Example Call - This call will create a secondary DNS IP Set:

```
./dnsmeapi.pl https://api.dnsmadeeasy.com/V2.0/dns/secondary/ipSet -X
POST -H accept:application/json -H content-type:application/json -d
'{"name":"IP New IP Set","ips":["1.1.1.1"]}'
```

Example Output:

```
{"name":"New IP Set","id":12345,"ips":["1.1.1.1"]}
```

IP Set Delete – Delete an IP Set

Example Call- Delete an IP Set with ID 12345:

```
./dnsmeapi.pl
https://api.dnsmadeeasy.com/V2.0/dns/secondary/ipSet/11341/ -X DELETE
```

Example Error:

```
{"error":["IP set is currently in use and cannot be deleted."]}
```

The IP Set is currently assigned to a secondary DNS domain and must be removed prior to deletion.

Resources & Methods

The following resources and methods are available with the DNS Made Easy API:

/dns/managed – **operate on multiple domains for your account**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns a list of all domain names for your account.
DELETE	200 – OK	Deletes all domains for your account
PUT	200 – OK	Updates multiple domains based on the identifiers in the ids field. The following values can be updated for all domains provided in the list: • Global Traffic Director (Boolean)

		• Applied Template (numeric ID) • Vanity NS Config (numeric ID) • Custom SOA Record (numeric ID) • Zone Transfer (numeric ID) • Folder (numeric ID)
POST	201 – OK	Creates multiple domains based on a list of names provided within a MultiDomain object.

/dns/managed/{domainId} **– operate on a single domain**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns the domain object representation of the specified domain.
DELETE	200 – OK 404 – specified domain name is not found	Deletes the specified domain **WARNING: This is irreversible!**
PUT	200 – OK	Updates a domains based on the domainId identifier in the path. Pass in a Domain object in XML or JSON with the new values.

/dns/managed/{domainId}/records **– operate on multiple records for one domain**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns the record object representation of the records for the specified domain. The following are URL parameters that may be added to determine the data returned: • type – Record type. Values: A, AAAA, CNAME, HTTPRED, MX, NS, PTR, SRV, TXT • rows – Number of rows returns • page – The page number of records, based on the number of rows returned
DELETE	200 – OK	Deletes the specified records using a list of record identifiers provided in XML or JSON format for the given template. **WARNING: This is irreversible!**

POST	201 – OK	Creates a series of records for the given domain using the list of record information provided in XML or JSON format

/dns/managed/{domainId}/records/{recordId} **– operate on a record in a domain**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns the record object representation of the specified domain.
DELETE	200 – OK 404 – specified record name and type is not found	Deletes the specified record **WARNING: This is irreversible!**
PUT	200 – OK	Updates a record based on the recordId identifier in the path. Pass in a record object in XML or JSON with the new values.

/dns/managed/{domainId}/multi/records/ **– operate on a multiple records in a domain**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns the record object representations
DELETE	200 – OK 404 – specified record name and type is not found	Deletes the specified records **WARNING: This is irreversible!**
PUT	200 – OK	Updates records based on the recordId identifiers in the path. Pass in record objects in XML or JSON with the new values.
POST	201 – OK	Creates a series of records for the given domain using

		the list of record information provided in XML or JSON format

/dns/secondary **– operate on multiple secondary domains for your account**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns a list of all domain names for your account.
DELETE	200 – OK	Deletes all domains for your account
PUT	200 – OK	Updates multiple domains based on the identifiers in the ids field. The following values can be updated for all domains provided in the list: • Global Traffic Director • Applied Template • Vanity NS Config • Custom SOA Record • Zone Transfer • Folder
POST	201 – OK	Creates multiple domains based on a list of names provided within a MultiDomain object.

/dns/secondary/{domainId} **– operate on a single secondary domain**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns the domain object representation of the specified domain.
DELETE	200 – OK 404 – specified domain name is not found	Deletes the specified domain **WARNING: This is irreversible!**
PUT	200 – OK	Updates a domains based on the domainId identifier in the path. Pass in a Domain object in XML or JSON with the new values.

/dns/secondary/{domainId}/records **– operate on multiple records for** one secondary domain

Method	HTTP Status Codes	Description
GET	200 – OK	Returns the record object representation of the records for the specified domain. The following are URL parameters that may be added to determine the data returned: • type – Record type. Values: A, AAAA, CNAME, HTTPRED, MX, NS, PTR, SRV, TXT • rows – Number of rows returns • page – The page number of records, based on the number of rows returned
DELETE	200 – OK	Deletes the specified records using a list of record identifiers provided in XML or JSON format for the given template. **WARNING: This is irreversible!**
POST	201 – OK	Creates a series of records for the given domain using the list of record information provided in XML or JSON format

/dns/managed/{templateId}/records **– operate on multiple records for one template**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns record object representations of the records for the specified template. The following are URL parameters that may be added to determine the data returned: • type – Record type. Values: A, AAAA, CNAME, HTTPRED, MX, NS, PTR, SRV, TXT • rows – Number of rows returned • page – The page number of records, based on the number of rows returned
DELETE	200 – OK 404 – specified domain name is not found	Deletes the specified records using a list of record identifiers provided in XML or JSON format for the given template. **WARNING: This is irreversible and affects the**

		records for any domain associated with the given template!
PUT	200 - OK	Replaces the current set of records for a template using the provided list of records provided in XML or JSON format for the given template. **WARNING: This is irreversible and affects the records for any domain associated with the given template!**
POST	201 – OK	Creates a record for the given template using the record information provided in XML or JSON format

/dns/managed/{templateId}/records/{recordId} **– operate on one record for one template**

Method	HTTP Status Codes	Description
DELETE	200 – OK 404 – specified domain name is not found	Deletes the specified record for the given template. **WARNING: This is irreversible and affects the records for any domain associated with the given template!**
PUT	200 – OK	Updates a record for the given template using the record information provided in XML or JSON format

/usageApi/queriesApi **– get account usage information by year and month**

Method	HTTP Status Codes	Description
GET	200 – OK	Returns a list of QueryUsage objects.

/usageApi/queriesApi/{year}/{month} **– get account usage information for a given year and month**

Method	HTTP Status Codes	Description

GET	200 – OK	Returns a list of QueryUsage objects.

/usageApi/queriesApi/{year}/{month}/managed/{domainId} –
get usage information for a given year and month for one domain

Method	HTTP Status Codes	Description
GET	200 – OK	Returns a list of QueryUsage objects.

/usageApi/queriesApi/{year}/{month}/secondary/{domainId} –
get usage information for a given year and month for one secondary domain

Method	HTTP Status Codes	Description
GET	200 – OK	Returns a list of QueryUsage objects.

Error Reporting

Generic Bad API Request Error

This is a generic error response code which indicates an error in the request issued, typically formatting or syntax related.

```
<html><head><title>Apache Tomcat/7.0.12 - Error
report</title><style><!--H1 {font-family:Tahoma,Arial,sans-
serif;color:white;background-color:#525D76;font-size:22px;} H2 {font-
family:Tahoma,Arial,sans-serif;color:white;background-
color:#525D76;font-size:16px;} H3 {font-family:Tahoma,Arial,sans-
serif;color:white;background-color:#525D76;font-size:14px;} BODY
{font-family:Tahoma,Arial,sans-serif;color:black;background-
color:white;} B {font-family:Tahoma,Arial,sans-
serif;color:white;background-color:#525D76;} P {font-
family:Tahoma,Arial,sans-serif;background:white;color:black;font-
size:12px;}A {color : black;}A.name {color : black;}HR {color :
#525D76;}--></style> </head><body><h1>HTTP Status 404 - Not
Found</h1><HR size="1" noshade="noshade"><p><b>type</b> Status
report</p><p><b>message</b> <u>Not Found</u></p><p><b>description</b>
<u>The requested resource (Not Found) is not available.</u></p><HR
size="1" noshade="noshade"><h3>Apache Tomcat/7.0.12</h3></body></html>
```

HTTP 403 - Forbidden Error

Requests made with invalid credentials or an invalid *x-dnsme-requestDate* value will receive an **HTTP 403 – Forbidden response**.

Every request sent using the API includes a request date header (set by your computers current time). An example would be:

```
> x-dnsme-requestDate:Tue, 01 Jan 2013 01:10:17 GMT
```

DNS Made Easy responds with a header that includes a Date (set by our globally synchronized clocks). An example would be:

```
< Date: Tue, 01 Jan 2013 01:10:17 GMT
```

If the date/time of the system issuing the API calls is 30 seconds or more off from the API servers date/time this will cause this error. The system time of the server issuing the API calls should be set correctly to prevent this.

Here is a full example that shows this:

```
./dnsmeapi.pl -v
http://api.dnsmadeeasy.com/V2.0/dns/managed/123456/records -X GET

* About to connect() to api.dnsmadeeasy.com port 80 (#0)

*   Trying 208.94.147.111... connected

* Connected to api.dnsmadeeasy.com (208.94.147.111) port 80 (#0)

> GET /V2.0/dns/managed/1234567/records HTTP/1.1

> User-Agent: curl

> Host: api.dnsmadeeasy.com

> Accept: */*

> x-dnsme-apiKey:*********************************

> x-dnsme-hmac:**********************************

> x-dnsme-requestDate:Fri, 25 Jul 2014 12:37:47 GMT

>
```

```
< HTTP/1.1 200 OK

< Server: Apache-Coyote/1.1

< x-dnsme-requestId: *******************************

< x-dnsme-requestsRemaining: 148

< x-dnsme-requestLimit: 150

< Set-Cookie: *****************************************

< Content-Type: application/json

< Transfer-Encoding: chunked

< Date: Fri, 25 Jul 2014 12:37:47 GMT

<

* Connection #0 to host api.dnsmadeeasy.com left intact

* Closing connection #0

{"data":[{"name":"","value":"1.1.1.1","id":15562953,"type":"A","source
":1,"gtdLocation":"DEFAULT","failed":false,"failover":false,"sourceId"
:1234567,"monitor":false,"dynamicDns":false,"ttl":3600,"hardLink":fals
e}],"page":0,"totalPages":1,"totalRecords":1}
```