# Technical Writing Samples

Katie Tothill

June 8, 2022

# Table of Contents

# Introduction

The following writing samples comprise my professional technical writing portfolio. These samples are designed to demonstrate my ability to write internal and external technical documentation. The upstream (internal) examples are intended for Domain Name System (DNS) engineers and support analysts. The downstream (external) examples are intended for users that have DNS service with the fictional company, “OurDNS”. This document does not define best practices for any type of documentation.

# Chapter 1. Internal

## 1.1. Bash Script

These scripts generate DNS query reports for a single account or a list of accounts. When aggregated, these reports produce a single report for a given date range. These reports are used by Account Managers and should not be provided to customers.

*Single Account Procedure*

1. Gather the following account data to generate reports for a single account:

| Parameter | Example Value |
|---|---|
| Account ID | 12345 |
| Date Range | 01/01/2020-12/31/2020 |
| Monthly Query Limit | 1,000,000 |
| Network | Managed DNS (represented here as network 0) |

2. Run the following script to generate a daily query report:

```
$ query-logs daily-report --account 12345 --start 01/01/2020 --end 12/31/2020
DAILY_REPORT_12345
```

3. Run the following script to aggregate the daily data for a given timeframe:

```
$ query-logs growth-report --daily-report "DAILY_REPORT_OUTPUT_12345" --query-limit
"1,000,000" --networks 0 "GROWTH_REPORT_OUTPUT_12345"
```

*Multiple Account Procedure*

Use the following script when generating query reports for a list of accounts.

**IMPORTANT** Each account ID and query limit are unique.

1. Create a list of account IDs in a text file, such as `account_ids.txt`.
2. Run the following script to generate a daily query report using the `account_ids.txt` file:

```
$ for aid in $(cat account_ids.txt); do query-logs daily-report --account ${aid}
--start 01/01/2020 --end 12/31/2020 DAILY_REPORT _${aid}; done
```

3. Run the following script to aggregate the daily data for a given timeframe:

```
$ declare -a aid_queries=(
"8156,4600000000"
```

```
"9742,3500000000"
"3332,60000000000"
"1401,500000000"
"8724,15000000000"
"6014,1500000000"
"12530,3000000000"
)

for aid_query in ${aid_queries[@]}; do aid=$(awk -F',' '{print $1}' <<< ${aid_query});
query=$(awk -F',' '{print $2}' <<< ${aid_query}); query-logs growth-report --daily
-report "DAILY_REPORT_OUTPUT_${aid}" --query-limit "${query}" --networks 0
"GROWTH_REPORT_OUTPUT_${aid}"; done`
```

# 1.2. Case Studies

*Introduction*

I resolved the following customer issues using the "Situation, Task, Action, Result" (STAR) framework.

## 1.2.1. Managed DNS

*Situation*

Customer experienced resolution delays of as much as 15-20 minutes when provisioning a new hostname via the API. Queries were prevented from receiving a correct response until the minimum Time to Live (TTL) elapsed. Until TTL expiration, the name server returned a Non-Existent Domain (NXDOMAIN) response.

*Task*

Identify and mitigate customer’s resolution delays and NXDOMAIN responses when provisioning a new hostname via the API.

*Action*

Review of sample publish call confirmed that the change had propagated to the edge and was available. Review of the customer’s provided code identified an incomplete task status value. The task status value identifies what phase the publish task is in. Once in a completed status, the hostname becomes live on the edge.

*Result*

Once the task status value was in a completed status, propagation was immediate.

## 1.2.2. Email Delivery

*Situation*

Customer received the following Simple Authentication Security Layer (SASL) error message while configuring a new server to send through the OurDNS email delivery system via Postfix: "SASL authentication failed". SASL provides mechanisms for authentication, data integrity-checking, and encryption for application development. This error message indicated a failed authentication

attempt when connecting to the OurDNS email server to send via Postfix.

*Task*

Identify and resolve authentication error message, so customer can connect to the OurDNS email server and send via Postfix.

*Action*

Reviewed customer configuration for both servers. Tested locally and determined that the "SASL authentication failed" error message was in response to the `libplain.so` or `liblogin.so` modules not installed in the `/usr/lib/sasl2` directory.

Locally tested a rehash of the SASL password for two servers. Determined that SASL passwords are required to be rehashed for each server and cannot be copied between them.

*Result*

Customer rehashed SASL and updated libraries for their OS and sent via Postfix successfully.

# Chapter 2. External

## 2.1. API Help Guide

*Introduction*

The following instructions describe how to review your zone details with a single API call. Previously, retrieving complete zone details via the OurDNS API required multiple calls. Should you still wish to utilize multiple API calls to gather zone details, see our "(Legacy) Get Zone Details" Help Guide.

*Prerequisites*

This table details the parameters to use when configuring the API call:

| **Required Parameters** | **Definition** |
|---|---|
| key | Your API key. **Note:** To view an existing key or generate a new key, see our "API Keys" Help Guide. |
| zone | Name of the zone you are retrieving hosts and records for. |
| **Optional Parameters** | **Definition** |
| FQDN | Fully Qualified Domain Name. If specified, the records returned are limited to records found at or beneath this host. |

*Procedure*

Use the `GET` API call and endpoint below to view the details for any zone and its corresponding records within your account:

```
$ curl -X GET -H "APIKEY:$key" https://api.ourdns.com/accounts/details/zone/FQDN
```

Example response:

```
{
  "id": "310422af9f792d37dffb528b",
  "hostmaster": "hostmaster@example.com",
  "ttl": 3600,
  "nx_ttl": 3600,
  "retry": 7200,
  "zone": "example.com",
  "refresh": 43200,
  "expiry": 1209600,
  "dns_servers": [
    "ns1.p01.ourdns.com",
    "ns2.p01.ourdns.com",
    "ns3.p01.ourdns.com",
    "ns4.p01.ourdns.com"
  ],
```

```
  "networks": [0],
  "network_pools": ["p01"],
  "primary": {
    "enabled": false,
    "secondaries": []
  },
  "records": [
    {
      "id": "310022af9f782d37dffb1790",
      "type": "NS",
      "ttl": 3600,
  "short_records": [
    "ns1.p01.ourdns.com",
    "ns2.p01.ourdns.com",
    "ns3.p01.ourdns.com",
    "ns4.p01.ourdns.com"
      ],
      "domain": "example.com"
    },
    {
      "id": "310512509f782d58bb1df419",
      "type": "A",
      "ttl": 3600,
      "short_records": [
        "1.2.3.4"
      ],
      "domain": "www.example.com"
    }
  ],
  "meta": {}
{
```

## 2.2. Sub-Delegation Help Guide

This OurDNS Help Guide details the steps to separate zone ownership between a parent zone and child zone, often referred to as sub-delegation.

A common use case for sub-delegation would be to delegate responsibility for a segment of the DNS name to a subset of users or another DNS provider. For example, the domain `ourdns.com` includes `help.ourdns.com` as a child zone whose management is restricted to the team managing this Help Guide.

| **NOTE** | For technical specifications, please see RFC 1034 Domain Concepts and Facilities |
|---|---|

*Procedure*

1. Add the following NS records to the child zone on **our** platform. Creating these NS records assigns the child zone to the other provider.

| Zone | Host | Record Type | Records |
|---|---|---|---|
| example.com | child.example.com | NS | ns1.p01.otherdns.com |
| | | | ns2.p01.otherdns.com |
| | | | ns3.p01.otherdns.com |
| | | | ns4.p01.otherdns.com |

2. Once the above records have been added to the child zone on our platform, create `child.example.com` at your other DNS provider as a parent zone with your desired records as usual.
3. After sub-delegation is configured successfully, requests for `example.com` and its associated hosts would still be answered by us; requests for `child.example.com` and its associated hosts would be answered by the name servers of your other DNS provider.
4. An example request for hosts at `child.example.com` would be answered as follows:

`Root Name Servers → .com Name Servers → example.com Name Servers (Us) → child.example.com Name Servers (Other DNS Provider) → Requested Host(s) and Record(s)`

*Testing Sub-Delegation*

Should you wish to test this configuration, we recommend using `dig` as described in our "Using Dig to Confirm DNS Changes" Help Guide.

A example of a successful `dig` is shown here:

```
$ dig child.example.com +trace
```

Example response:

```
; <<>> DiG 9.10.6 <<>> child.example.com +trace
;; global options: +cmd
.           85384   IN  NS  a.root-servers.net.
.           85384   IN  NS  b.root-servers.net.
.           85384   IN  NS  c.root-servers.net.
.           85384   IN  NS  d.root-servers.net.
.           85384   IN  NS  e.root-servers.net.
.           85384   IN  NS  f.root-servers.net.
.           85384   IN  NS  g.root-servers.net.
.           85384   IN  NS  h.root-servers.net.
.           85384   IN  NS  i.root-servers.net.
.           85384   IN  NS  j.root-servers.net.
.           85384   IN  NS  k.root-servers.net.
.           85384   IN  NS  l.root-servers.net.
.           85384   IN  NS  m.root-servers.net.
;; Received 239 bytes from 8.8.8.8#53(8.8.8.8) in 46 ms

com.            172800  IN  NS  e.gtld-servers.net.
com.            172800  IN  NS  g.gtld-servers.net.
```

```
com.            172800  IN  NS  l.gtld-servers.net.
com.            172800  IN  NS  b.gtld-servers.net.
com.            172800  IN  NS  d.gtld-servers.net.
com.            172800  IN  NS  a.gtld-servers.net.
com.            172800  IN  NS  i.gtld-servers.net.
com.            172800  IN  NS  m.gtld-servers.net.
com.            172800  IN  NS  c.gtld-servers.net.
com.            172800  IN  NS  k.gtld-servers.net.
com.            172800  IN  NS  h.gtld-servers.net.
com.            172800  IN  NS  j.gtld-servers.net.
com.            172800  IN  NS  f.gtld-servers.net.
;; Received 844 bytes from 192.36.148.17#53(i.root-servers.net) in 27 ms

example.com.    172800  IN  NS  ns1.p01.ourdns.com.
example.com.    172800  IN  NS  ns2.p01.ourdns.com.
example.com.    172800  IN  NS  ns3.p01.ourdns.com.
example.com.    172800  IN  NS  ns4.p01.ourdns.com.
;; Received 296 bytes from 192.5.6.30#53(a.gtld-servers.net) in 22 ms

child.example.com.  172800  IN  NS  ns1.p01.otherdns.com.
child.example.com.  172800  IN  NS  ns2.p01.otherdns.com.
child.example.com.  172800  IN  NS  ns3.p01.otherdns.com.
child.example.com.  172800  IN  NS  ns4.p01.otherdns.com.
;; Received 256 bytes from 198.51.45.8#53 (ns2.p01.ourdns.com) in 27 ms

child.example.com.  3600    IN  A   23.227.38.65
;; Received 64 bytes from 198.51.43.8#53(ns2.p01.otherdns.com) in 21 ms
```

**NOTE**

If you are attempting to delegate an existing child zone to another DNS provider, you may need to contact `support@ourdns.com` to confirm that these changes will not adversely impact DNS resolution. If you are creating a brand-new child zone, no extra steps are required.

# Conclusion

Thank you for taking the time to review my provided samples. I hope they provided valuable insight into some of my skills and abilities. Have a charming day!