dnsimple

Go API Cheatsheet

With the DNSimple library you can easily interact our powerful API to administer domain names, configure DNS records, provision and install SSL certificates, and more.

Getting Started

1. Install the Go library

```
go get github.com/dnsimple/dnsimple-go/dnsimple
```

2. Authenticate

Obtain your API access token: https://support.dnsimple.com/articles/api-access-token/

```
oauthToken := os.Getenv("TOKEN")
ts := oauth2.StaticTokenSource(&oauth2.Token{AccessToken: oauthToken})
tc := oauth2.NewClient(context.Background(), ts)

client := dnsimple.NewClient(tc)
client.BaseURL = "https://api.sandbox.dnsimple.com"
```

3. Check Authorization

If you want to know which account is associated with the current access token, you can use `#identity`. The account ID is required for the majority of API operations.

```
whoamiResponse, err := client.Identity.Whoami(context.Background())
accountId := strconv.FormatInt(whoamiResponse.Data.Account.ID, 10)

fmt.Printf("#{accountId} (your account ID)")

=> 1234 (your account ID)
```

Managing Domains

Check Domain Availability

Check if a domain is available for registration.

```
checkDomainResponse, err := client.Registrar.CheckDomain(
    context.Background(), accountId, "foo.com")

fmt.Printf("#{checkDomainResponse.Data.Available}")

=> true
```

Register A Domain

1. To register a domain, you need to specify a registrant_id. This can be fetched via the Contacts API.

```
contacts, err := client.Contacts.ListContacts(context.Background(),
    accountId, nil)
contact := contacts.Data[0]

fmt.Printf("#{contact.ID}")

=> 123
```

2. You can register the domain with this information.

```
registerDomainResponse, err := client.Registrar.RegisterDomain(
    context.Background(), accountId, "foo.com",
    &dnsimple.DomainRegisterRequest{RegistrantID: contact.ID})
registration := registerDomainResponse.Data

fmt.Printf("State: #{registration.State}\nAuto Renew:
           #{registration.AutoRenew}\nWhois Privacy:
           #{registration.PrivateWhois}\nRegistrant:
           #{registration.RegistrantID}"

=>State: registered
Auto Renew: false
Whois Privacy: false
Registrant: 123
```

DNS

Create a DNS record

Create a DNS A record to map an IP address to a domain.

```
recordValues := ZoneRecordAttributes{Name: String("www"),
                Content: "127.0.0.1", Type: "A"}

response, err := client.Zones.CreateRecord(context.Background(),
                         accountID, "foo.com", recordValues)
record := response.Data

fmt.Printf("#{record.ID}")

=> 123
```

Update a DNS record

Update a previously created DNS record.

```
recordValues := ZoneRecordAttributes{Ttl: 60}

response, err := client.Zones.UpdateRecord(context.Background(),
                         accountID, "foo.com", 5, recordValues)

updated := response.Data

fmt.Printf(updated.Ttl)

=> 60
```

SSL Certificates

Order an SSL Certificate with Let's Encrypt

Creates the purchase order. Use the ID to issue the certificate.

```
attributes := LetsencryptCertificateAttributes{}
response, err := client.Certificates.PurchaseLetsencryptCertificate(
                    context.Background(), accountID, "foo.com",
                    certificateAttributes)
cert := response.Data

fmt.Printf("ID: #{cert.ID}\nState: #{cert.state}")

=>ID: 123
State: new
```

Issue an Let's Encrypt Certificate

Issues the pending order. This process is async. A successful response means that the response is queued.

```
response, err := client.Certificates.IssueLetsencryptCertificate(
                    context.Background(), accountID, "foo.com", cert.ID)
certificate := response.Data

fmt.Printf("State: #{certificate.state}")
=>State: requesting
```

Install the certificate

Download the certificate.

```
response, err := client.Certificates.DownloadCertificate(
                    context.Background(), accountID, "foo.com", cert.ID)
cert := response.Data

file, err := os.Create("www_foo_com.pem")
defer file.Close()
_, err2 := file.WriteString(cert.Server)
_, err3 := file.WriteString("\n")
_, err4 := file.WriteString(strings.Join(cert.Chain,"\n")
```

Download the certficate's private key.

```
response, err := client.Certificates.GetCertificatePrivateKey(
                    context.Background(), accountID, "foo.com", cert.ID)
key := response.Data

file, err := os.Create("www_foo_com.key")
defer file.Close()
_, err2 := file.WriteString(key.PrivateKey)
```