dnsimple

C# API Cheatsheet

With the DNSimple package 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 package

```
dotnet add package DNSimple
```

2. Authenticate

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

```
using dnsimple;

var client = new Client();
client.AddCredentials(new OAuth2Credentials("your-api-token"));
```

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.

```
var identity = client.Identity.Whoami().Data

Console.WriteLine(identity.Account.Id);
=> 1234 (your account ID)
```

Managing Domains

Check Domain Availability

Check if a domain is available for registration.

```
var check = client.Registrar.CheckDomain(accountId, "foo.com");

Console.WriteLine(check.Data.Available);
=> true
```

Register A Domain

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

```
var contacts = client.Contacts.ListContacts(accountId);

Console.WriteLine(contacts.Data.First().Id);
=> 123
```

2. You can register the domain with this information.

```
var registration = client.Registrar.RegisterDomain(
    accountId,
    "foo.com",
    new DomainRegistrationInput{RegistrantId = contactId}
).Data

Console.WriteLine("State: {0}, AutoRenew: {1}, WhoisPrivacy: {2},
  Period: {3}, RegistrantId: {4}", registration.State,
  registration.AutoRenew, registration.WhoisPrivacy,
  registration.Period);
=> State: registered, AutoRenew: false, WhoisPrivacy: false,
   Period: 1, RegistrantId: 123
```

DNS

Create a DNS record

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

```
var zoneRecordInput = new ZoneRecord
{
    Name = "www",
    Content = "127.0.0.1",
    Type = ZoneRecordType.A,
};

var record = client.Zones.CreateZoneRecord(accountId, "foo.com",
             zoneRecordInput).Data;

Console.WriteLine(record.Id);
=> 123
```

Update a DNS record

Update a previously created DNS record.

```
var zoneRecord = new ZoneRecord {
    Ttl = 60
};
var updated = client.Zones.UpdateZoneRecord(accountId, "foo.com",
              record.Id, zoneRecord).Data;

Console.WriteLine(updated.Ttl);
=> 60
```

SSL Certificates

Order an SSL Certificate with Let's Encrypt

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

```
var attributes = new LetsencryptCertificateAttributes {
  AutoRenew = false,
  Name = "SuperCertificate"
};

var cert = client.Certificates.PurchaseLetsencryptCertificate(accountId,
           "foo.com", attributes).Data;

Console.WriteLine("Id: {0}, CommonName: {1}, AuthorityIdentifier: {2}",
                  cert.Id, cert.CommonName, cert.AuthorityIdentifier);
=> Id: 123, CommonName: www.foo.com, AuthorityIdentifier: letsencrypt
```

Issue an Let's Encrypt Certificate

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

```
var certificate = client.Certificates.IssueLetsencryptCertificate(
                  accountId, "foo.com", cert.Id).Data;

Console.WriteLine(certificate.State);
=> "requesting"
```

Install the certificate

Download the certificate.

```
var cert = client.Certificates.DownloadCertificate(accountId, "foo.com",
           certificate.Id).Data;

var chain = String.Join("/n", cert.IntermediateCertificates.ToArray());
string[] lines =
{
    cert.ServerCertificate, chain
};

await File.WriteAllLinesAsync("www_foo_com.pem", lines);
```

Download the certficate's private key.

```
var cert = client.Certificates.DownloadCertificate(accountId, "foo.com",
           certificate.Id).Data;

await File.WriteAllTextAsync("www_foo_com.key", cert.PrivateKey);
```