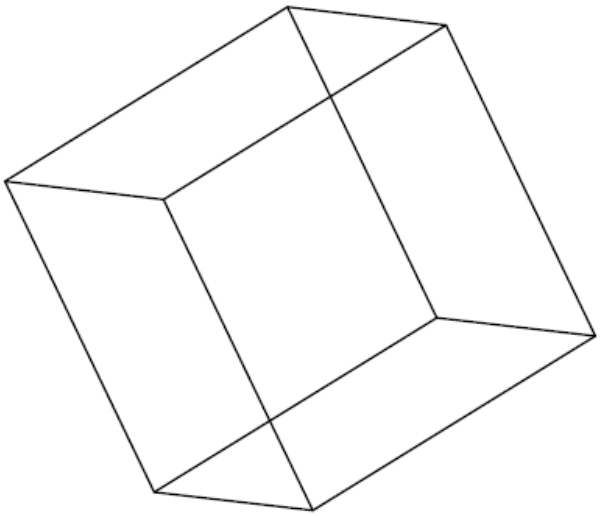
Miraclebox



Developer Guide
Part 5 – API
Release version 1.0

Document control

Version	Date	Action	Author
1.0	May 2022	Document published	Sam Wells

This document is versioned and stored in GitHub.

Related documentation

Related documentation is available on GitHub.

Resource	Location
Reference Documentation	http://github.com/miraclebox/documentation/ref/
GitHub code repo	http://github.com/miraclebox/application/api/*
WSDL Files	http://github.com/miraclebox/application/api/WSDL/*

Table of contents

Intended audience

This guide is for technical developers and covers API connection and usage. It is not an end user guide.

Document purpose

This document provides:

- Background information on what the API is and what it does
- A technical reference guide to the central API
- Instructions on how to get set up to use and call the API
- Sample code that can be used to call the API from within an application
- Support details

Introduction

Welcome to this developer guide to Miraclebox. Miraclebox is designed to reduce the development overheads inherent in building and managing campaign and competition microsites for clients:

- Provides a reporting portal so that technical involvement is no longer required for report production. This is now self-service for account managers and other end users. Account managers will be able to self-serve these reports in raw data and chart downloads. Ultimately this function will be enabled for clients to access directly

- Reduces development time and effort to set up sites. All data is now collected and managed through integration to storage and logic via API calls, rather than through code replication and new database setups. Additionally, services are fully tested and reused therefore test effort is greatly reduced
- Allows charging based on usage

About this guide

This guide covers the following

- Introduction and background to the service and relevant architectures
- How to connect to the service for CRUD operations

This guide does not cover

- End user functions
- UI usage
- Reporting and associated APIs
- Logging access. This is part of standard application architecture and the Azure portal

Prerequisite knowledge

To use this guide and the service, it is assumed developers are familiar with:

- Code
- Architecture
- API modes and terminology
- XML and its derivatives
- General networking concepts
- Technical tools
- This guide

Product overview

Description and features

Miracle box is a service platform and user portal designed to address inefficiencies in campaign microsite build and operation. Though they may look different functionally these sites for the most part functionally identical. Miraclebox replaces the need for developers to repeatedly build similar functions, database, and associated code, and to be involved in report generation. This has several advantages:

- Reduces potential for errors
- Reduces effort in build and test
- Reduces management overheads

Access to core functionality is enable via secure APIs that are called from applications.

Miraclebox is targeted at campaign and competition microsites, though any project that requires the provided functionality could be supported.

The portal enables self-service setup of:

- Clients
- Brands
- Campaigns
- Competitions
- Unique pack code series

Reporting can be conducted on:

- User entries
- Competition entries
- Competition wins
- Standard set of web analytics

Report data may be downloaded as a csv pack, or charts as .png files.

The need for any MVC model development is removed. Core developer functions are covered later in this document but include API calls for:

- End user set up of clients, brands, campaigns, and competitions
- Reporting
- Authorisation

What it does not do

Miraclebox does not remove the need for any development whatsoever. MVC UI and controller development is still required.

Restrictions

No functionality is externally available. This is an IT-imposed security restriction. Direct access cannot be given to clients or third parties.

Dependencies

The following are required for developer use and access:

- User account on the service platform
- Access to GitHub
- Browser with web access
- Local installs of
    - Eclipse IDE
    - SOAPUI client
    - A text editor

Details are covered in Getting Started

Use-case Examples

1. Find all competition entries for campaigns during a date range
2. Find all unique site visits for a specific brand
3. Find the tail for campaigns x and y and plot a line chart of this data

Known limitations

Access is only from within miracle Corp network infrastructure on Azure

Where to get support

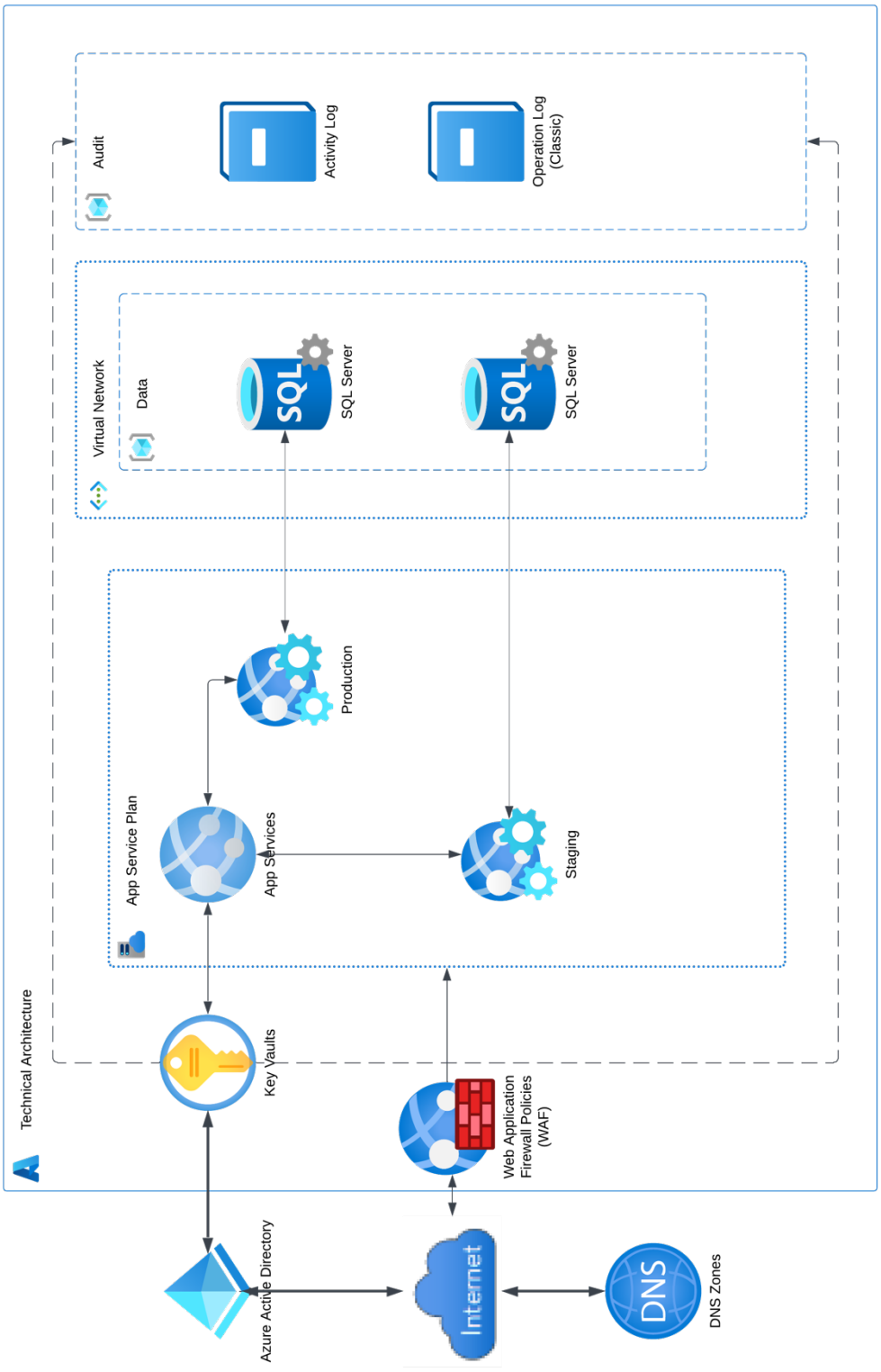
miraclesupport@org.com

Development effort

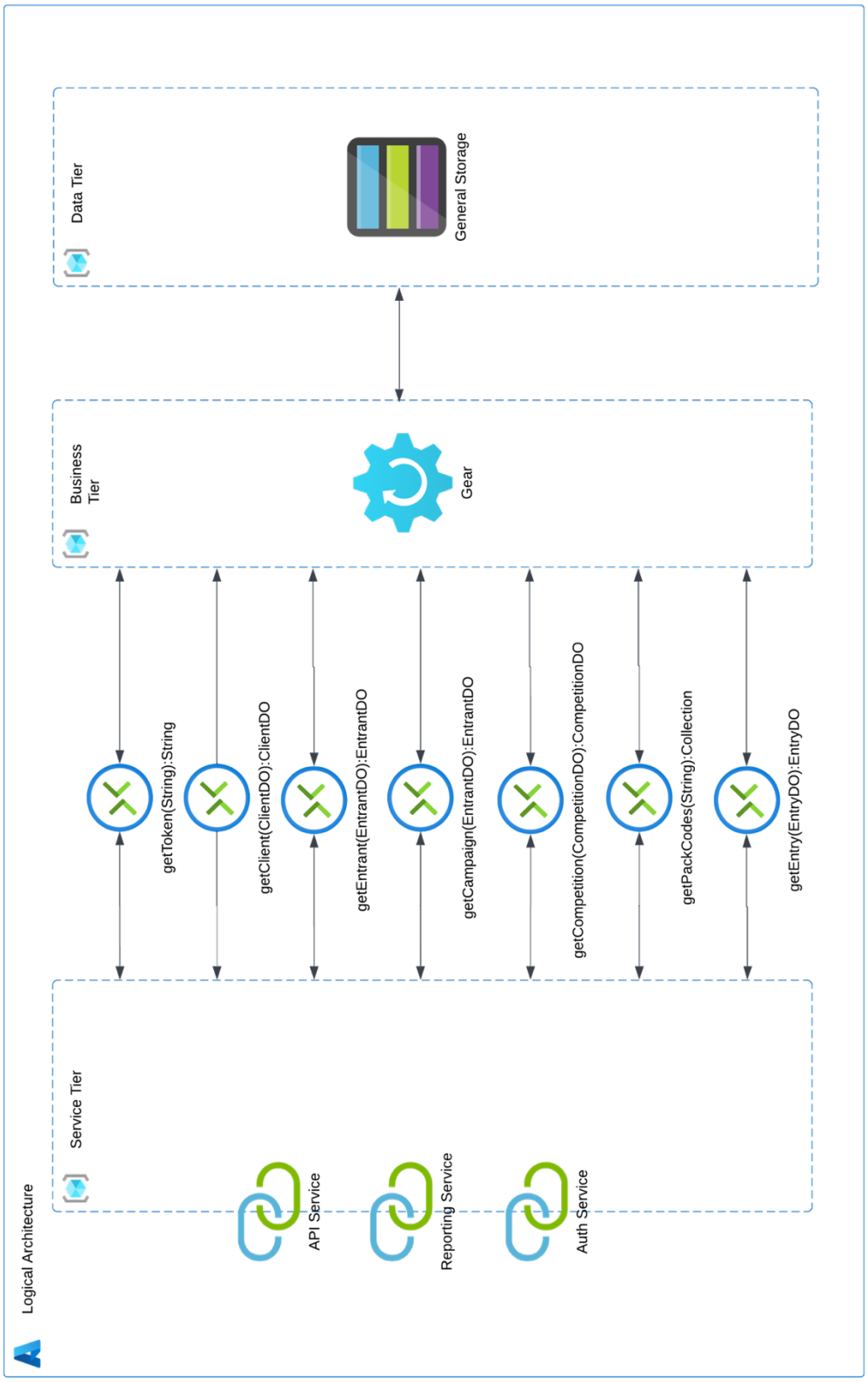
Budget 2 hours per call including unit testing but excluding integration test.

Components

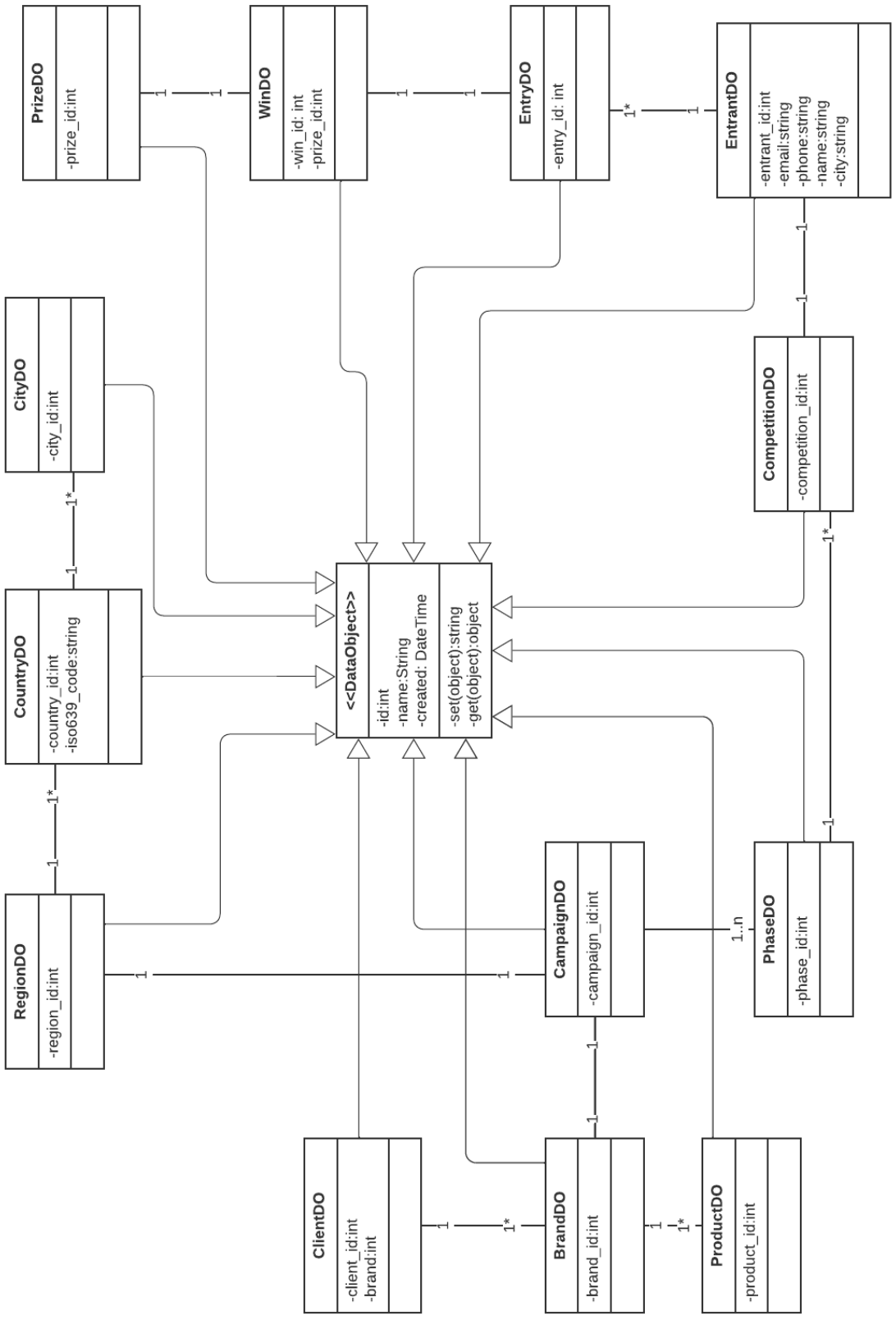
Technical Architecture



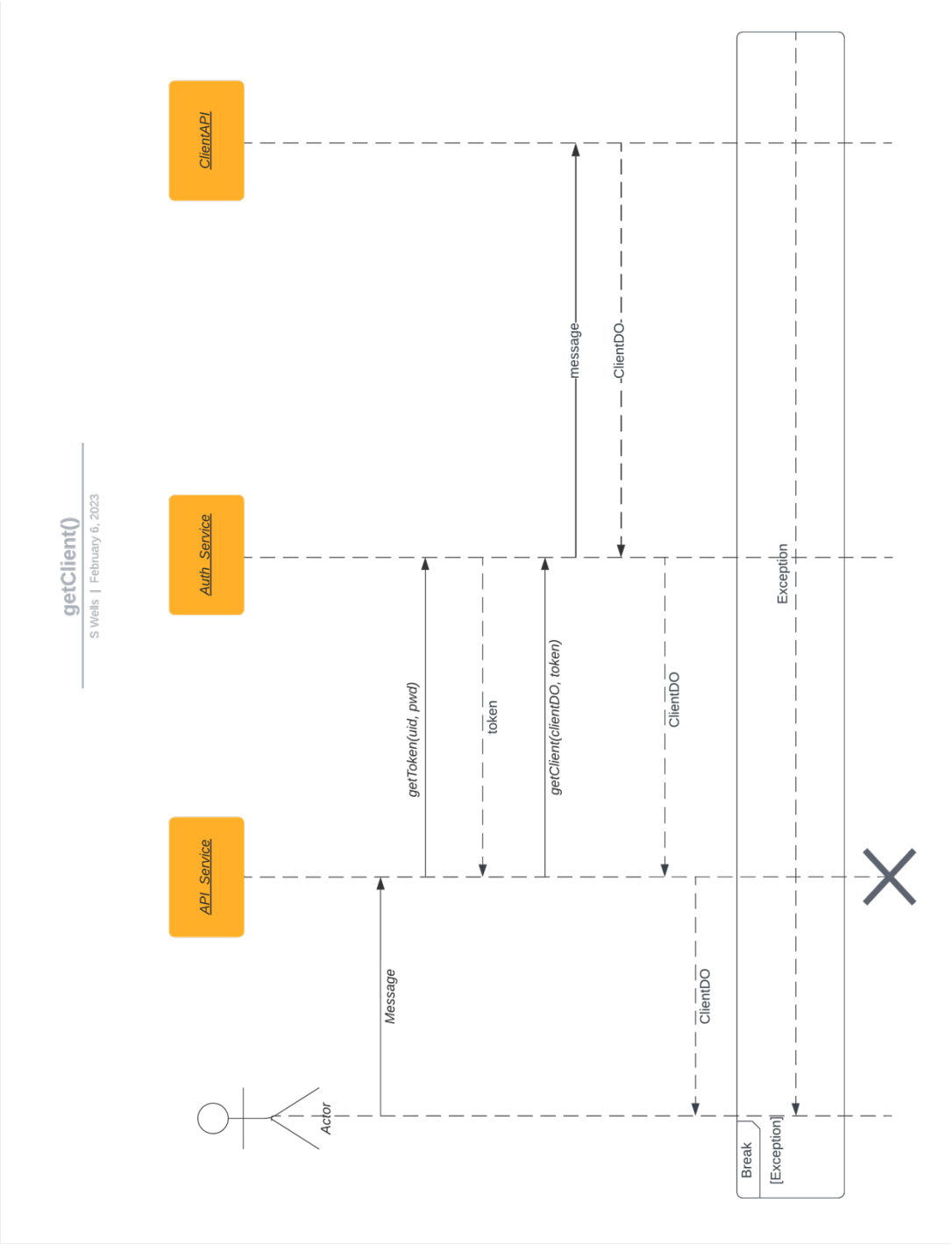
Logical Architecture



Data Object Model



API Workflow

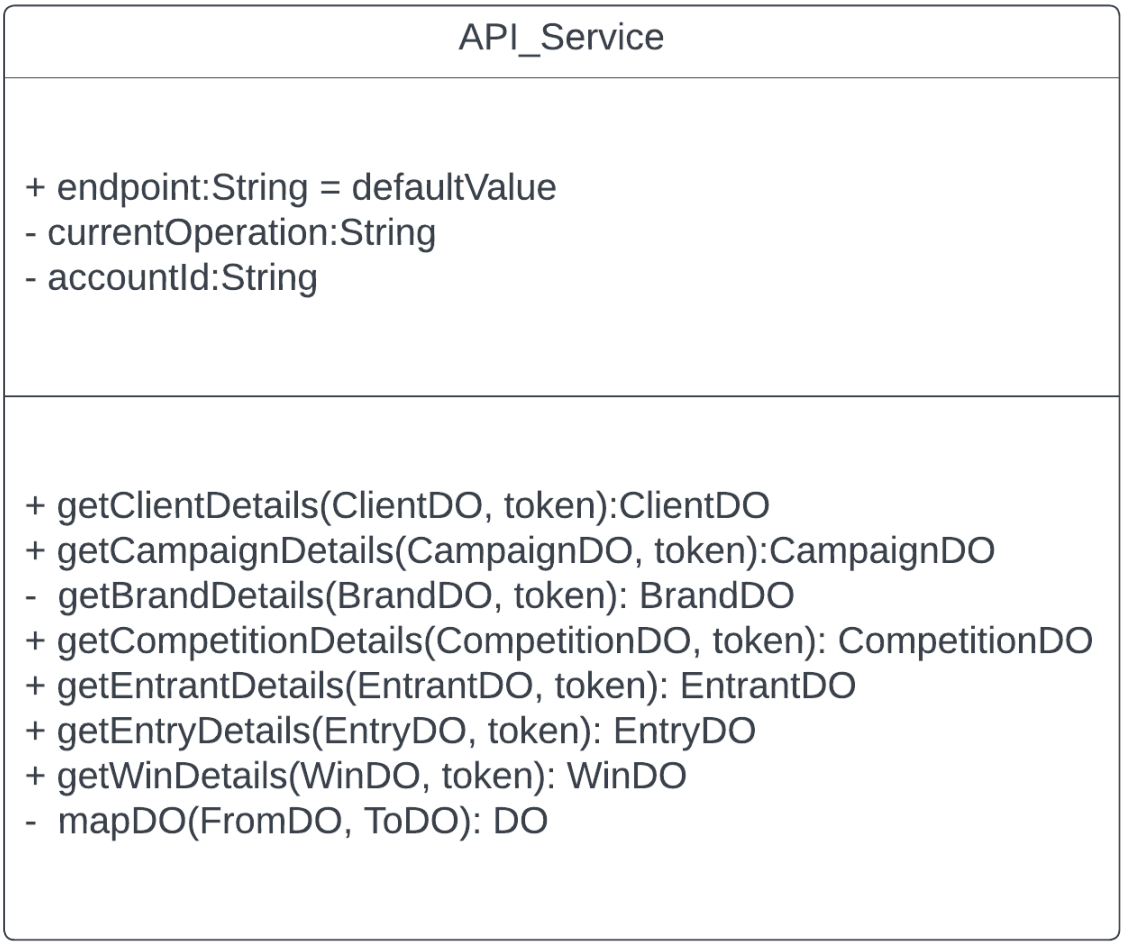


Architecture Notes

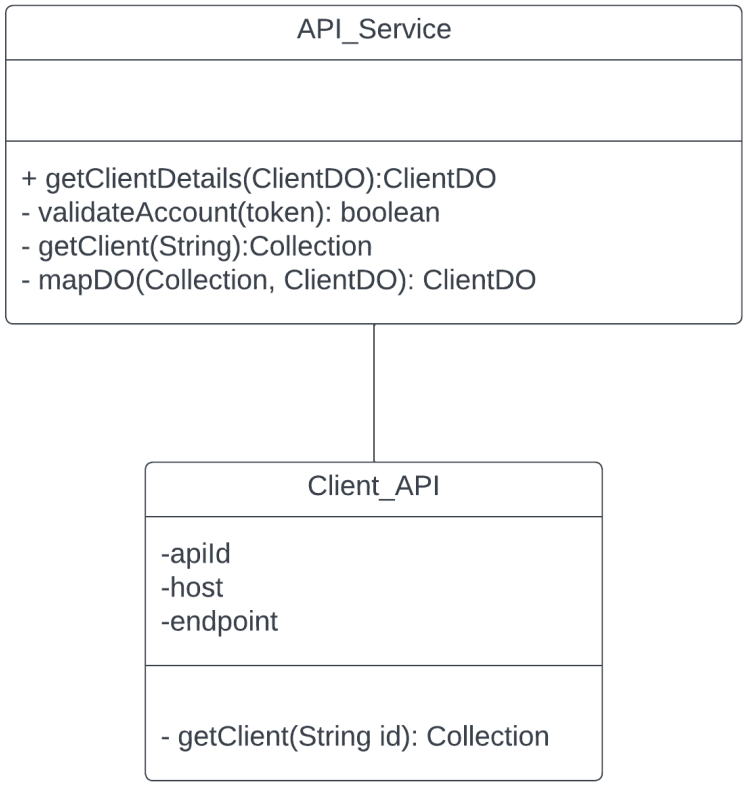
1. The workflow for one API call is shown. However, all calls follow the same pattern
2. The first call in an API access workflow will always be to the `getToken()` method. This is managed by `API_Service`. However, credentials must be provided

Getting started

The approach described is sanctioned architecturally. In production, the APIs must be accessed using a java library. Within this library the following class is provided:



API_Service utilises a Class per type of API. ClientAPI is shown here for context only:



The classes may be found in the `miraclebox.integration.api.*` package

Credentials and security

Calls can be made only by registered applications, from within specific server infrastructures and IP address ranges. These include all miracle Corp IP address ranges and Azure infrastructures.

Accounts must be authorised prior to use. To request authorised account details please contact:

getmiracle@miraclecorp.com

The process can take several days, so it is recommended this be done as soon as possible.

Once access is confirmed, proceed to *Setting Up*

Set up a test call

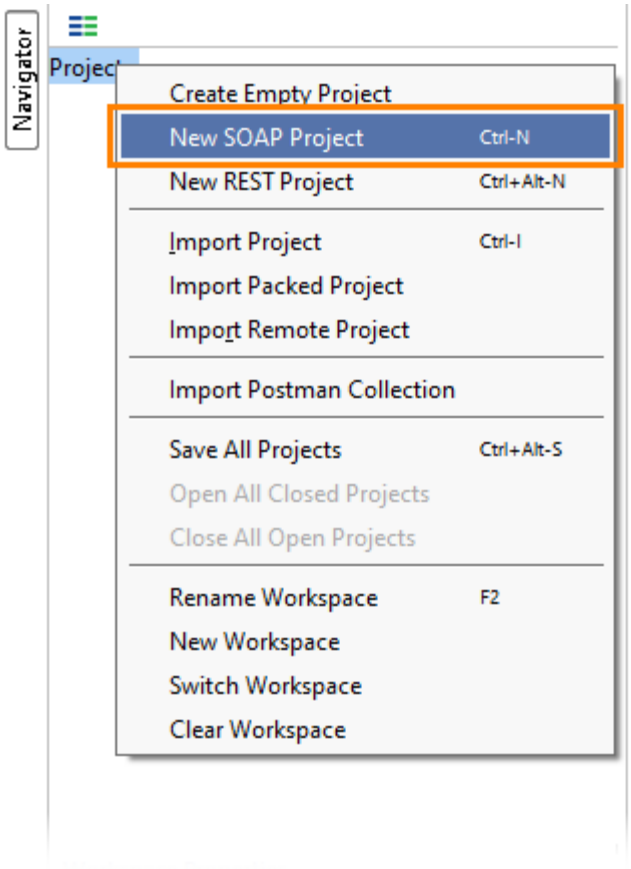
Download and install SOAPUI

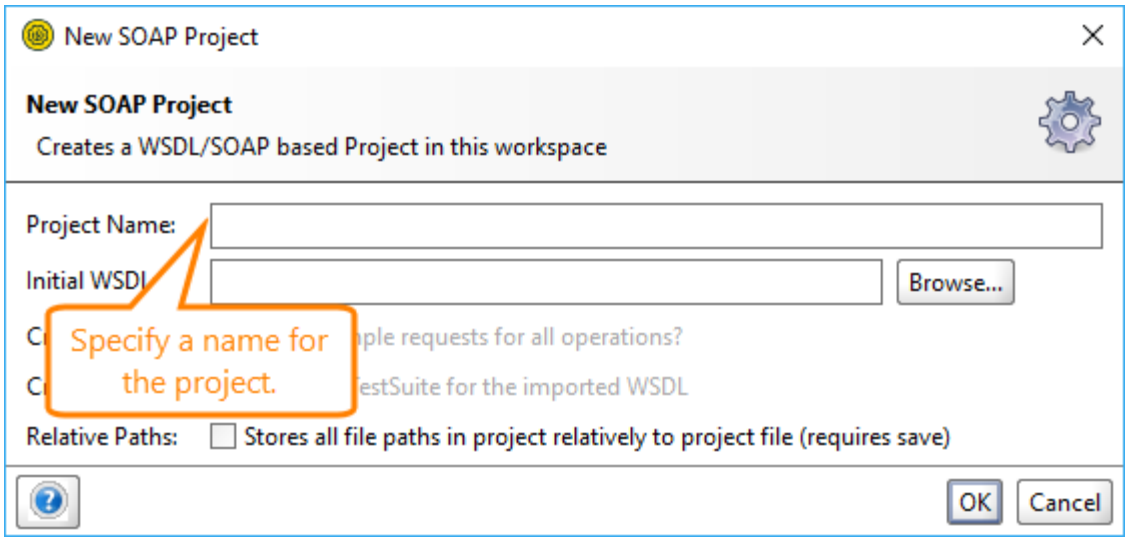
We use SOAPUI to send and test soap requests. SOAP UI is open source and free. Please download version x here:

https://www.soapui.org/downloads/soapui/

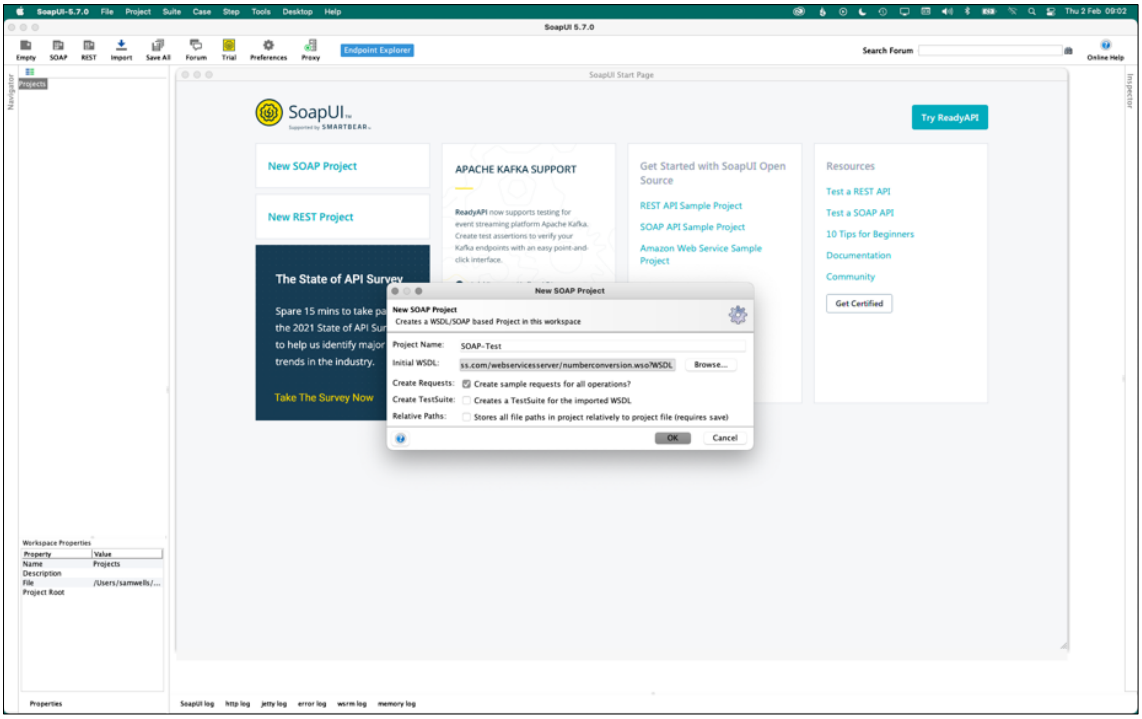
Launch the file and follow the steps indicated. Once installed move on to *Create a Project*.

Create a project

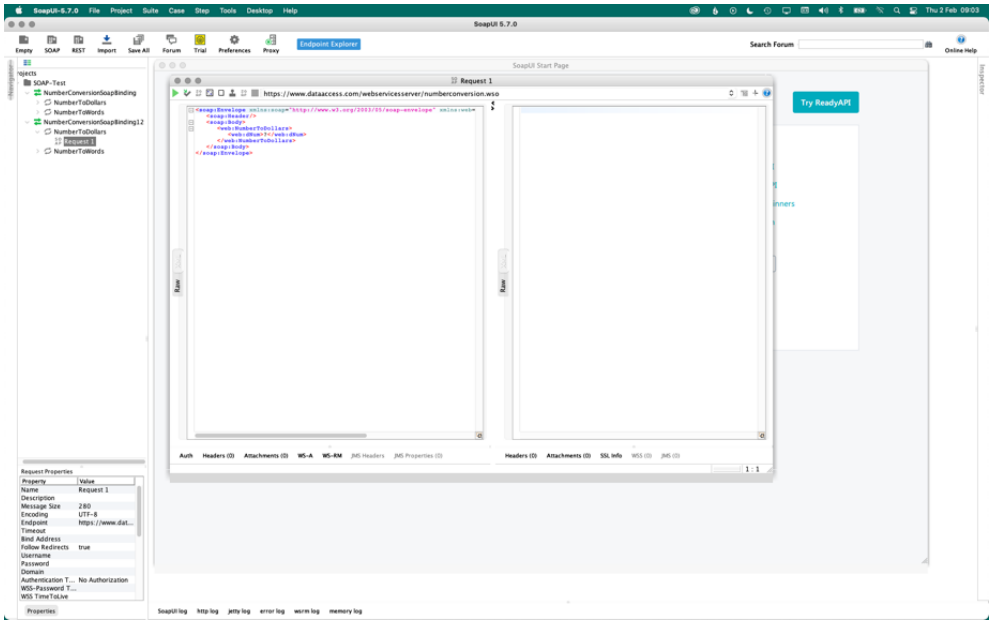




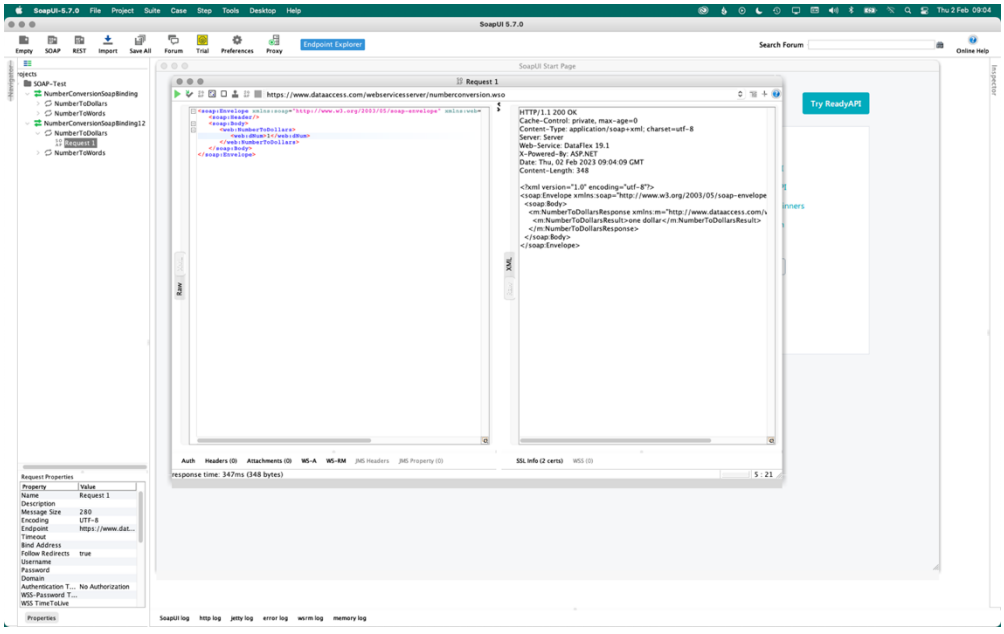
Add the WSDL file



Prepare request data



Send the Request and evaluate the response

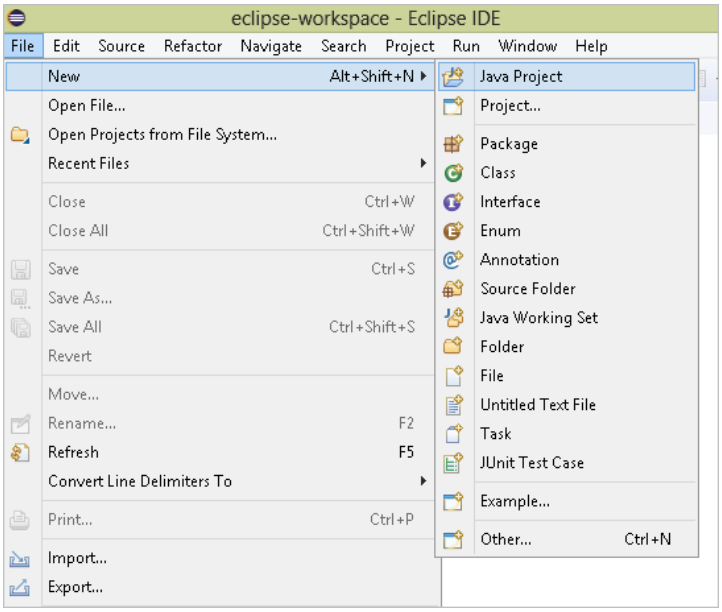


The response will appear in the right-hand window. The header code will indicate the status.

Set up a call from code

For this walkthrough we will set up java code to call `getClientDetails()` on the Miraclebox service.

Set up a new project in Eclipse



Set up the Java code

The API is exposed in Java as a collection of methods on the API_Service class described in this section.

Clone the GitHub repo

http://github.com/miraclebox/application/api/*

- This repo includes the required WSDL file
- Ensure the code is available in Eclipse

Add a new source file within the test package

`Miraclebox.api.test/api.java`

Add imports

```
import miraclebox.data.dataobjects;
import miraclebox.api.API_Service;
```

Add constructor

```
API_Service tuesday = new APIService();
```

Prepare ClientDO object

```
ClientDO client = new ClientDO();
Client.setId('5000');
```

Add method

```
callClient(String client) returns ClientDO {

     client = tuesday.getClientDetails(client);
     System.out.println(client).

}
```

Check the code and compile.

The API_Service class handles the API connection within the ClientAPI Class.
Run this code and evaluate the response in the console. You should see the client details output to the console within Eclipse. The actual API request and response can be seen in the http log.

Example Request


```
POST /xml/getClientDetails.asmx HTTP/1.1 Host: www.website.com
Content-Type: application/soap+xml Content-Length: 349

<?xml version="1.0" encoding="utf-8"?> <soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
<soap:Body> <GetClient> xmlns="https://www.website.com/xml/">
<UserId>75</UserId> </GetClient> </soap:Body> </soap:Envelope>
```


Example Response

HTTP/1.1 200 OK Content-Type: application/soap+xml; charset=utf-8 Content-Length: 413 <?xml version="1.0" encoding="utf-8"?> <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body> <GetClientDetails xmlns="https://www.website.com/xml/"> <GetClientResult><id>5000</id><name>EvilCorpInc</name><brand><id>15</id><name>RaisinFlakes</name></brand> </GetClientResult> </GetClientResponse> </soap:Body> </soap:Envelope>

Try running with alternative parameters in the request.

Error Codes

There are no application specific response codes other than those detailed below.

1. HTTP response codes
2. Java Exceptions

RecordNotFoundException	A data error indicating a search for the record with the provided details returned no results.
NoAccountException	Valid credentials have not been supplied
APIOfflineException	The API is not available. Contact Systems Admin

Troubleshooting

Most issues will be the result of invalid code or Exceptions. For anything outside of this please contact support in the first instance.

End of document

