# ArtMaps Platform

## Introduction

The ArtMaps platform is a collection of client and server applications developed to support two interrelated activities:

1. Crowd-sourcing of geographic (quantitative) information about an [Object of Interest](#GlossaryObjectOfInterest) (OoI)
2. Crowd-sourcing of qualitative information about an OoI using [Semi-Structured Blogging](#GlossarySemiStructuredBlogging).

The current applications that make up the platform are:

* A [Windows Azure SQL Database](http://www.windowsazure.com/en-us/home/features/data-management/) – Provides data storage for OoIs and related data.
* A [Windows Azure Cloud Service](http://www.windowsazure.com/en-us/home/features/cloud-services/) – Provides programmatic access to data stored in the SQL database via a ReSTful API.
* A [WordPress](http://wordpress.org/) installation – Provides a web interface to the stored data as well as managing user authentication and the process of semi-structured blogging.
* An [iOS](http://www.apple.com/iphone/ios/) application – Provides a mobile interface to the stored data.

[Figure 1](#Figure1) below shows these applications and the flow of communication between them.

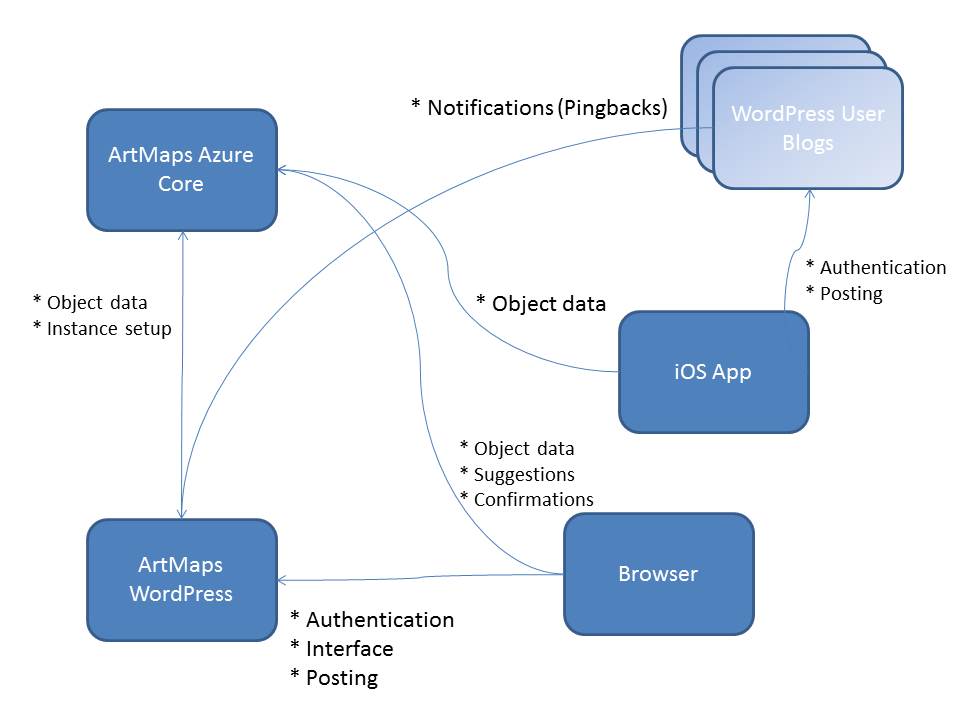


Figure 1

## Compilation and Installation

### Requirements

* A [Windows Azure](http://www.windowsazure.com/en-us/) account with SQL Database and Cloud Service capabilities activated.
* A multi-user [WordPress](http://wordpress.org/) installation, this is not just a WordPress.com account but a WordPress installation that you must have full administrative rights over.
* [Visual Studio](http://www.microsoft.com/visualstudio/eng/visual-studio-update) with the [.net Azure SDK](http://www.windowsazure.com/en-us/develop/downloads/) installed for compiling the ArtMaps Azure cloud service. The codebase has been tested with Visual Studio 2012 and version 1.8 of the Azure SDK.
* [SQL Server Management Studio](http://www.microsoft.com/en-gb/download/details.aspx?id=29062) for setting up the SQL database schema.
* A copy of the source code available on [github](https://github.com/horizon-institute/artmaps).
* A version of [OpenSSL](http://www.openssl.org/) installed.

### Creating and Initialising the Database

1. Open SQL Server Management Studio (SSMS) and connect to your SQL Database instance.
2. Create a new database within that instance and make a note of the name.
3. Within SSMS, open the *AzureCore/Schema/ArtMapsCoreAzureSchemaV2.sql* from the code checked out from github. Executing this SQL script will build the schema within the database.
4. Create a new database user (or use an existing user that is trusted) and assign them full permissions for both the database and its associated federation(s). Make a note of the username and password.

### Configuring and Deploying the AzureCore Service

1. From the repository, open *AzureCore/AzureCore.sln* file in Visual Studio.
2. In Visual Studio, open AzureCore -> Roles -> UI and choose the settings tab, the following values need to be set:
   * **ArtMaps.DevFabric.Tracing.Path**When the service is run in the development fabric, specifying a directory path here will cause log files to be saved on the path.
   * **ArtMaps.Diagnostics.BufferQuota**Diagnostic log buffer quota in megabytes, a sensible default is 50.
   * **ArtMaps.Diagnostics.LogLevel**Diagnostic logging level, valid values are *Undefined*, *Critical*, *Error*, *Warning*, *Information* and *Verbose*.
   * **ArtMaps.Diagnostics.TransferPeriod**  
     Period, in minutes, for transferral of diagnostic logs to Azure storage. A sensible default is 60.
   * **ArtMaps.Security.KeySize**Size, in bits, for RSA security keys generated by the service. A sensible default is 512.
   * **ArtMaps.SqlServer.ConnectionString**Connection string for the ArtMaps database instance
   * **ArtMaps.Storage.ConnectionString**Set this to an SQL Azure storage account connection string.
   * **ArtMaps.Storage.Containers**Leave blank, this can be used to initialise blob containers that the service uses, however the service at present does not use any blob containers.
   * **ArtMaps.Storage.Queues**Leave blank, this can be used to initialise queues that the service uses, however the service at present does not use any queues.
   * **ArtMaps.Storage.Tables**Leave blank, this can be used to initialise tables that the service uses, however the service at present does not use any tables.
   * **Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString**Set this to an SQL Azure storage account connection string.
   * **Microsoft.WindowsAzure.Plugins.Caching.ConfigStoreConnectionString**As above.
3. Configuration is now complete but before deployment an RSA private key needs to be generated. Using OpenSSL, generate a new RSA private key (recommended key size is 2048 bits) and save it in PEM format in *AzureCore/AzureCore/UIContent/Keys* with the filename *MasterKey.pem*. Following this, the key also needs to be cloned and stored in a blob format compatible with the .NET Cryptographic Service Provider. A utility application for performing this task is included in the repository under *PemToCspBlob*. The project will need to be opened in Visual Studio and built before it can be used. The application runs on the command line and the syntax is *PemToCspBlog –i <PEM input> -o <Blob output>*. The blob formatted key needs to be saved under *AzureCore/AzureCore/UIContent/Keys* with the filename *MasterKey.blob*.
4. The service is now ready to be published.

### Configuring and Deploying the WordPress Plugin and Theme

The WordPress plugin and theme do not need configuring or compiling before deployment. To deploy, copy the folder *WordPress/plugin/artmaps* from the repository to your WordPress installation’s plugin directory and the folder *WordPress/theme/artmaps* to the theme directory. From the *Network Admin* console in WordPress you will need to activate the ArtMaps plugin for the whole network and choose the ArtMaps theme as the default theme. The theme depends on the WordPress 2012 theme being present (this is the default theme in WordPress 3.4 which the plugin/theme have been developed against).

At this point, the plugin needs to be configured at the WordPress network level, from the network admin dashboard, select the *ArtMaps Settings* option. There are four values that need to be configured:

1. Core Server URL: This is the base URL for the Azure core service, e.g. [*http://myartmapsservice.example.com*](http://myartmapsservice.example.com)
2. Master Key: This is the master key in PEM format that was generated during the configuration of the Azure core service. Copy and paste the content of the file in this field.
3. Google Maps API Key: This is a [Google maps API key](https://code.google.com/apis/console) that has been set up to allow API requests from the domain in which your WordPress instance is hosted.
4. IP InfoDB API Key: An API key for the [IPInfoDB service](http://ipinfodb.com/).

Once configured, any existing blogs in the WordPress multi-site instance will be converted to ArtMaps instances.

There are configuration settings for individual blogs however these do not initially need to be changed. The configuration settings are discussed in the architecture section below. The database can be quickly bootstrapped using a CSV file. From the blog dashboard select *ArtMaps Import*. The structure of the CSV file is important. The first 3 columns are latitude, longitude and a unique ID. Following this there may be a variable number of columns which each hold a metadata value. The metadata names are specified by the column headings (headings for the first 3 columns are ignore).

Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Latitude | Longitude | ID | Name | Shape |
| 50.1 | 0.8 | 13234 | Box | Square |

ArtMaps is now installed and ready to use, to reach the map page suffix your site URL with /map.

## Architecture

### Windows Azure SQL Database Schema

The schema diagram is shown in [Figure 2](#Figure2). A single SQL database instance is multi-tenanted, it supports multiple ArtMaps instances by logically isolating the data through the use of a partition key. In order to achieve this use has been made of [SQL Database Federations](http://msdn.microsoft.com/en-us/library/windowsazure/hh597452.aspx) each tenant is referred to as a *Context* in the ArtMaps application, *ContextID* columns is the federation key used in each federated schema table. The tables in the schema are:

* Action – Stores an action performed by a user in the *URI* column, *Actions* are documented later in this document.
* ActionLocation – Relates an *Action* to a *Location*.
* Context – The *Contexts*, or tenants, that exist in the schema. *Keys* are documented later in this document.
* Location – A ‘wrapper’ around different the different location types (elliptical, named, polygonal and point locations).
* LocationEllipse – Stores a geographical location defined by an ellipse. Currently not in use.
* LocationNamed - Stores a geographical location defined by an address. Currently not in use.
* LocationNamedPart – The parts (lines) of an address.
* LocationPoint – Stores a geographical location defined by a point with a radius of error (equivalent to a reading from a GPS device).
* LocationPolygon – Stores a geographical location defined by a polygon. Currently not in use.
* ObjectMetadata – Metadata for an OoI stored as name/value pairs.
* ObjectOfInterest – An OoI, the URI is a unique reference for each object.
* Pingback – Stores Pingback information. Currently not in use.
* PingbackAction – Relates a pingback to an action. Currently not in use.
* PingbackLocation – Relates a pingback to a location. Currently not in use.
* PingbackObject – Relates a pingback to an object. Currently not in use.
* PingbackUser – Relates a pingback to a user. Currently not in use.
* Sequence – Primary key sequence table, used by the *NextID* stored procedure.
* User – Stores users.

The following stored procedures are also part of the schema:

* NextID – Call with the table name to get the next primary key for that table.
* SelectObjectWithinBounds – Searches for OoIs within geographic bounds.
* SelectObjectWithinBounds2 – As above but for version 2 (still in development) of the API.



Figure 2

### Windows Azure Core Service

The core service is a Windows Azure Cloud Service with one *Web Role*, the solution (as in the github repository) is separated into 7 projects which are:

* AzureCore – The Cloud Service project which contains configuration files and the signing keys.
* AzureUtilities – An F# project that contains utilities for working with Azure but ArtMaps independent code.
* Controllers – An F# project containing the Controllers for the Web Role.
* Core – An F# project containing ArtMaps specific code that does not rely on any web framework.
* Model – A C# project containing the Linq-to-SQL classes for the SQL Database schema.
* UI – An ASP.NET MVC 3 Web Role.
* Utilities – Generic utilities in F#.

### WordPress Plugin and Theme

### iOS Application

Todo

## Glossary

**Object of Interest**: A ‘thing’ which is the focus of crowd-sourcing operations, e.g. an artwork

**Semi-Structured Blogging**: A method of providing structure to blog posts that allows a blog user to post in a natural manner but aids in the automated processing of the post.