MYMIS – DEPLOYMENT MANUAL

Version 1.1 | November 2016

Document version	Date	Notes
1	06/10/2016	Initial document version.
2	29/11/2016	Add database password reset feature

Index

Deployment Solution

We have developed a deployment solution that performs the work of instantiating the structure necessary for the platform (see What artifacts are deployed?), as well as setting it up with the necessary contents.

This solution consists of an Azure Resource Manager template, available in our GitHub, at https://github.com/numbersbelieve/MyMIS-Deployment.

This solution attempts to simplify the deployment as much as possible, so that potential users will only have to configure a small number of parameters to get the platform to work. For advanced configurations, users should follow this manual, and edit the Azure resources' settings accordingly.

What artifacts are deployed?

The initial step in performing a deployment of the platform is to create the structure containing all of the necessary components. This is represented by an Azure Resource Group, containing the following:

- 1 SQL Server with 1 SQL Database;
- 1 App Service plan, containing two separate web apps:
  - 1 Node.js website ('Extensibility');
  - 1 ASP.NET website, containing the UI, API and command processing web job.
- 1 Storage (Tables, Blobs and Queues).

Optionally, the platform can also make use of:

- A SendGrid account, creatable from the Azure Portal.

What service levels are used in the artifacts?

- The SQL Database is created on an S0 level (minimum recommended, can increase as necessary)
- The App Service plan is created on the Basic level, which is the minimum possible due to some features we use in the web app. Standard level gives you the possibility of having staging slots.
- The storage is created on the Standard_LRS level (minimum possible, can increase as necessary).

How do I perform a deployment of the platform?

By accessing our github, or our deployment page, it is possible to begin the deployment process by pressing the Deploy to Azure button. This button will open the portal, and load our template into it, showing the user all the parameters they need to configure.

After configuring these parameters, it is possible to select the subscription and resource group to deploy the template into. Resource groups may be created in this step if they do not exist yet – all the resources are named based off the resource group's ID.

After reviewing and accepting Microsoft's legal terms, and pressing the Create button, the deployment process will begin. It takes between 5-10 minutes, and when it ends, the platform itself will begin its remaining setup processes.

These processes consist of preparing the database for access, as well as creating all of the necessary tables and containers in the storage, and loading initial data into these. As soon as they are completed, the platform is ready to use.

How do I test the deployment's success?

After the deployment is finished, by logging in with the new administration account, the user should be redirected to the Subscription Management section. Otherwise, the section is accessible via the user menu.

There, the user can create a new tenant, based on the Modeling Template. After tenant creation, access the Modeler (you will need to grant modeling permissions to the user if you are not using the administration account). From there, you can begin modeling applications on the myMIS platform.

Platform Management

How to add a certificate / How to map DNS?

For the platform to use a certificate, there are a series of steps:

1) Obtain the certificate you want to use, for example, from Let's Encrypt;
2) Upload it to the web app;
3) In its app settings, change the WEBSITE_LOAD_CERTIFICATES as well as the MyMis.Certificate.Thumbprint settings to match the new certificate's thumbprint.

If it is an encryption certificate only, this will be enough, as Azure will use its own certificate for the HTTPS. However, if you want to also use your own HTTPS:

4) Configure your DNS to point to the web app's *.azurewebsites.net URL, or its IP;
5) Add a Custom Domain to the web app;
6) Add an SSL binding to the web app in the SSL Certificates configuration area;
7) After DNS propagates, ensure your app's settings are pointing to the new URL (e.g. MyMis.Web.Endpoint should be https://www.example.com)
8) Ensure the OAuth configuration is pointing to the new DNS (Auth.Clients table)

How to add additional templates?

Additional templates may be obtained from our GitHub. For example, here are our tutorials, which contain templates with the result of the completed tutorial.

How to update the platform?

Automatic updates are not included as part of the deployment solution. If you are interested, contact us and we will develop an update strategy.

How to reset a Tenant database user/password?

If you want to reset a tenant database user or password, you can follow the steps:

1. Update the *DatabaseUser* column at the SQL Table *[Auth].[Tenants]* with the value ***WAITING_FOR_UPDATE*** for the tenant that you want;
2. Restart the Web App;
3. The user will be updated (or created if not exists) with the name *P[PLATFORM INSTANCE CODE]U[TENANT CODE]* and a new Database Password.
    a. *[PLATFORM INSTANCE CODE]* – Uses the value of *MyMis.PlatformInstanceCode* web app parameter.
    b. *[TENANT CODE]* – Uses the code for the given tenant.

Platform Configuration

How to add external logins?

The platform supports the ability to perform external logins with four different providers: LinkedIn, Facebook, Twitter and Microsoft, by configuring the web app's App Settings.

To configure these:

LinkedIn

Follow the procedures described on https://developer.linkedin.com/docs/oauth2. Input the API key and secret into **MyMis.ExternalAuth.LinkedIn.Key** and **MyMis.ExternalAuth.LinkedIn.Secret** (create if they don't exist).

Facebook

Create a new application on https://developers.facebook.com. Obtain the App ID and App Secret, and add them into **MyMis.ExternalAuth.Facebook.ID** and **MyMis.ExternalAuth.Facebook.Secret.**

Twitter

Create a new application on https://apps.twitter.com/. Set the callback URL to your application's callback URL, and input the API key and API secret into **MyMis.ExternalAuth.Twitter.Key** and **MyMis.ExternalAuth.Twitter.Secret** (create if they don't exist).

Microsoft

Create a new application on https://apps.dev.microsoft.com. Generate a password for it, and input the ID and that password into **MyMis.ExternalAuth.Microsoft.ID** and **MyMis.ExternalAuth.Microsoft.Secret**.

How to add Google Analytics?

The platform supports the sending of data to a Google Analytics subscription. See https://support.google.com/analytics/answer/1032385?hl=en to configure.

After obtaining the Tracking ID, input it into the **MyMis.Options.GoogleAnalytics** parameter in the app settings (create it if it doesn't exist). After rebooting, the platform will begin sending data to Analytics.

If there is any issue, you can use the Tag Assistant extension by Google to debug what is happening.

How to add administration users?

There are two configuration properties that we can fill to add additional administration users (e.g. to have access to statistics with an account that isn't the master account).

In **MyMis.Administration.Users**, we define the users to be created, separated by ';'. In **MyMis.AdministrationAccounts.Passwords**, we can define their passwords – if left blank, they will created with the default password 'adminadmin', which should be changed on the first login.

Appendixes

Web App Parameters

- **AzureWebJobsDashboard, AzureWebJobsStorage, MyMis.Storage.ConnectionString** – Connection strings to the platform storage;
- **MyMis.Options.HideGenericExceptions** – Set to false if you want users to see more detail on exceptions;
- **MyMis.Options.GoogleAnalytics** – Google Analytics tracking ID. See How to add Google Analytics?
- **MyMis.MasterAccount** – Email of the administrator of the installation. Account is created with the default password 'adminadmin', which should be changed;
- **MyMis.AdministrationAccounts.Users / .Passwords** – Configuration options for additional administration users. See How to add administration users?
- **MyMis.DefaultCompany** – Identifies an OEM – e.g. "mymis" will assume that the mymis OEM is the default one, for the initial landing page and composing URLs;
- **MyMis.Branding** – General branding parameter used throughout the platform.
- **MyMis.MaxFileSizeInMB** – Maximum file size to allow in uploads. Per file;
- **MyMis.PlatformInstanceCode** – Identifies an installation for a given customer. Can be used when the Customer has multiple platform deployments. The parameter has a maximum size of 60 characters. Can contain letters as defined in the Unicode Standard 3.2 and decimal numbers;
- **MyMis.SQL.*.ConnectionString** – Set of connection strings to connect to the SQL server. Deployment.ConnectionString can be deleted after the initial setup;
- **MyMis.OAuth2.*** - ID/Secret pair used to communicate. Can be changed, but needs to be changed in the database as well (Auth.Clients table);
- **MyMis.*.Endpoint** – Identify different URL targets for the application's different components. All except ExtensibilityEngine.Endpoint need to be changed when you change DNS (see How to add a certificate / How to map DNS?);
- **MyMis.ConnectorHub.*** – Connector configurations. Leave with the default values;
- **MyMis.Keys.Encrypt.PrivateKey** – Encryption key. Leave with the default value;
- **MyMis.Plugins.Directory** – Plugins location. Leave with the default value;
- **Microsoft.WindowsAzure.Plugins.*** - Configurations for a set of plugins. Leave with the default values;
- **MyMis.Certificate.Thumbprint / WEBSITE_LOAD_CERTIFICATES –** Thumbprint of the certificate you want to use. See How to add a certificate / How to map DNS?
- **MyMis.Notification.From –** "From" email to be seen in emails the platform sends;
- **MyMis.Notification.SendGrid.Credentials.* -** Parameters to authenticate on a valid SendGrid account.
- **MyMis.ExternalAuth.* -** See How to add external logins?
- **MyMis.SQL.Tenant.* -** Pointers to the database. Leave with the default value;

- **MyMis.DeploymentEnvironment –** Identifies the deployment environment. Leave with the default value;

Performing backups

The platform does not perform any backups on its own. If the subscription owner wants to perform their own backups, this section describes the necessary points to take under consideration.

For the database, the SQL Azure backup structure is ideal. It can generate .bacpac backups of the database, and maintain them in any storage.

The platform's blob storage is organized in the following way:

- Each tenant (identified by a GUID) has a set of folders:
  - **Commands** contains all the submitted commands in that tenant;
  - **Report-Containers** contains the .rdlc files;
  - **Scripts** contains the .js and .cs files;
  - **Binary** contains files that have been attached to the tenant in any context;
  - **Temporary** contains files that were necessary for download (calendar, csv, contact exports, PDFs downloaded from external systems…). Does not need to be backed up.
- **Connector** contains connector licenses;
- **Oem-brand** contains the branding images for different platform OEMs;
- **Tenant** contains the images associated to each tenant;
- **Tools** contains tools downloadable inside the platform. Does not need to be backed up;
- **User** contains the images associated to the users;
- **Wad-*** contains Windows Azure logs. Does not need to be backed up.

The table storage is used for logging information.

Essentially, to perform a backup of the structure behind a subscription, it is necessary to back up the database, the storage except for temporary folders and tools, and the table storage.

Logging information and command history may need to be saved only for a small amount of time – deleting them after a given amount of time is a good idea, as they will be the largest size of information in the backups.

Logging and debugging

There are multiple places you can obtain more information on your subscription, if you are having issues.

HideGenericExceptions

By setting MyMis.Options.HideGenericExceptions to false, you can obtain more details on exceptions.

Application logs

Both the application and the extensibility site have logs which you can choose via the Azure portal to log to a storage and/or filesystem. By accessing

websitename.scm.azurewebsites.net, you also have access to a set of tools, of which the most important are:

- Web job console: Allows you to see the commands that the web job is getting to process, as well as a full log of what it is doing.
- Logstream: Allows you to see the logs of the given website as a stream. After accessing, must send some form of data in order for the URL to load.

Deployment logs

By accessing the Azure portal, there is an option named Activity Log which displays logs on all operations performed within it. During the deployment process, you can follow this log to understand what operations are being performed, as well as identify any failures.

Error pages

By accessing the deployed website via FTP, you can edit the web.config files in both the API and UI sections of the application, and set the parameter “customErrors” to “Off”, from “RemoteOnly”. This will make it so that the platform displays stack traces of the errors, instead of a “sorry” page, when they happen.