# Getting Started with ReMi

ReMi (Release Management Interface) is a web based tool for planning and organising releases. ReMi consists of two separate applications: ReMi.Web – a single page application (SPA) and ReMi.Api – a .NET WebApi that uses a JSON RESTful API to expose functionality to the SPA client. The following steps show how to access the ReMi NuGet packages and set up a Visual Studio project so that you can access and extend the ReMi functionality.

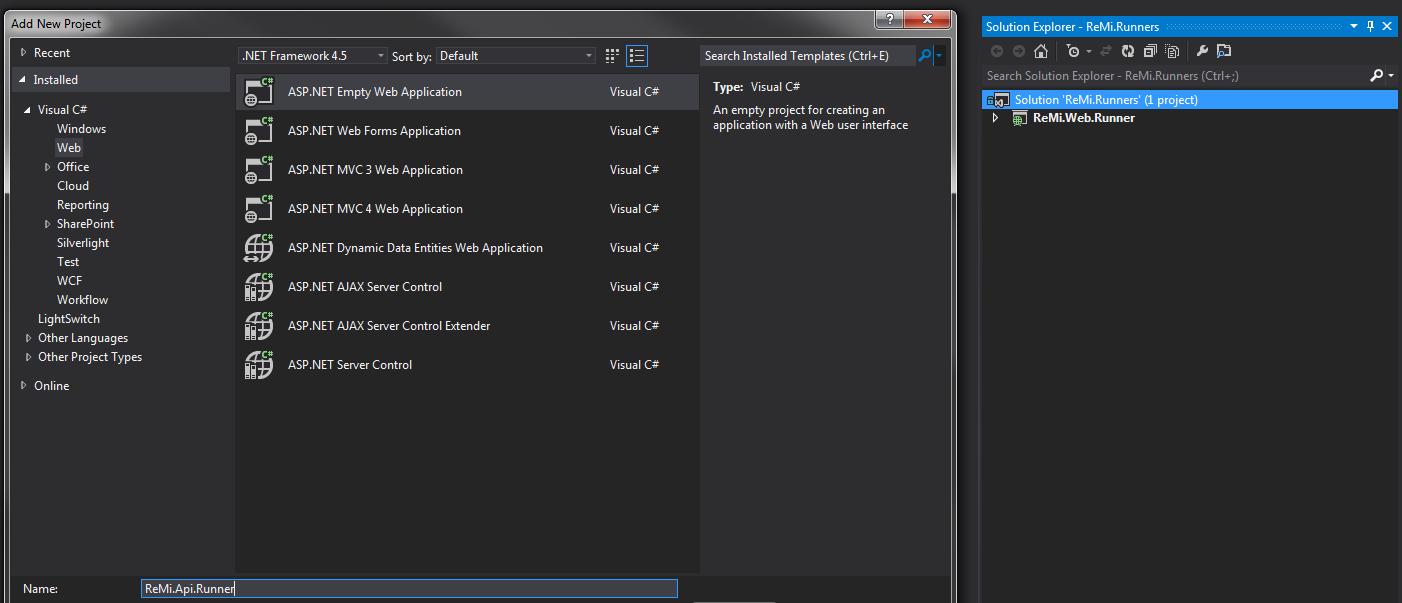
# Prerequisites

* Visual Studio 2012 or newer
* MS SQL 2008 R2 or newer
* IIS 7.5 or newer with ASP.NET 4.0 installed

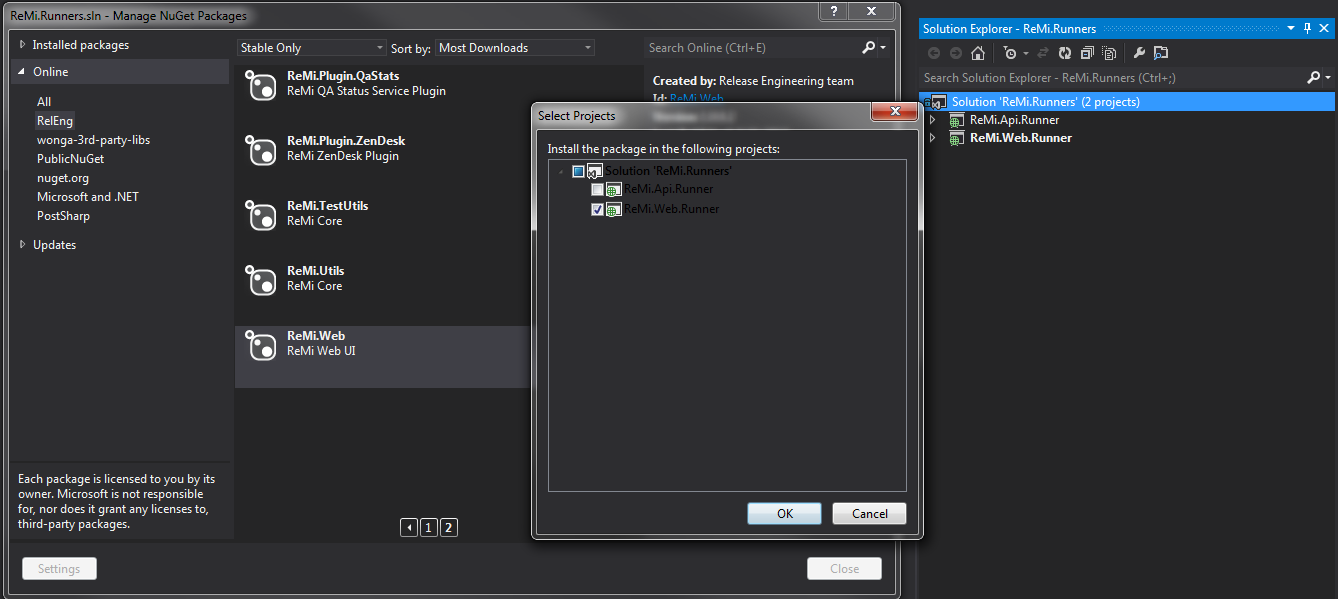
# Creating the Visual Studio project

In this example we will create a Visual Studio solution that will contain the packages for the ReMi web API and the ReMi web client application. We will use the name ‘ReMi.Web.Runner’ for the web application, ‘ReMi.Api.Runner’ for the API and ‘ReMi.Runners’ as the solution name.

1. In Visual Studio create a new empty ASP .Net Web Application named ‘ReMi.Web.Runner’, ensure that the Solution name is set to ‘ReMi.Runners’ before clicking on the ‘Ok’ button:



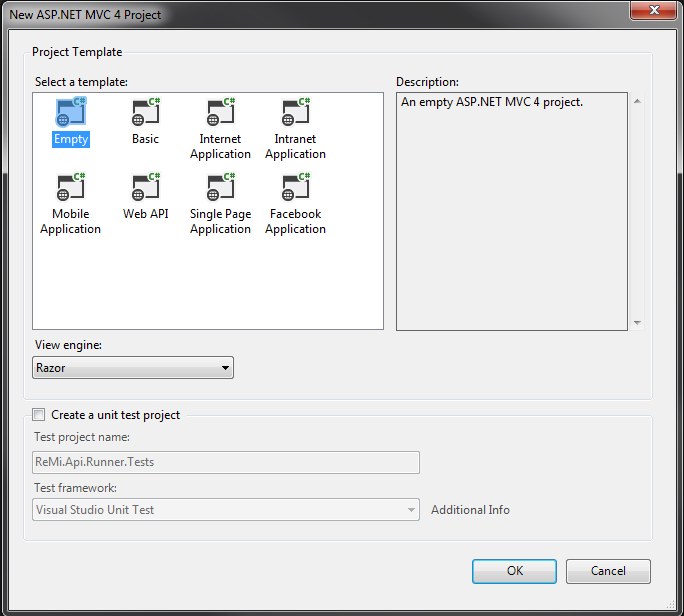
1. Install the ReMi.Web package from NuGet UI package manager:



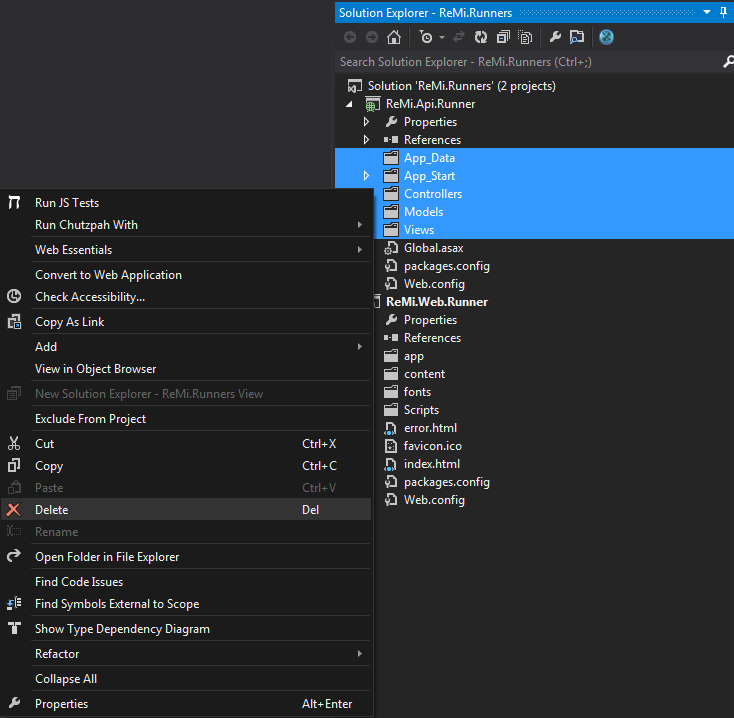
Or alternatively run the following command from the Package Manager Console:

**Install-Package –ProjectName ReMi.Web.Runner ReMi.Web**

1. Add an empty ASP.NET MVC 4 Web Application for the API and name it for ‘ReMi.Api.Runner’:



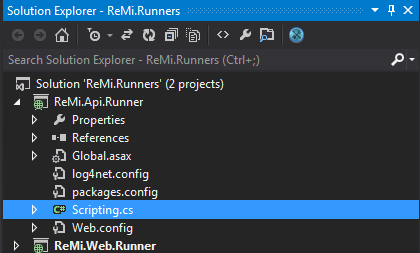
1. Remove all of the folders created in the step above



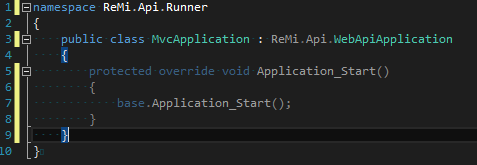
1. Now install the following packages for the ReMi API:

**Install-Package –ProjectName ReMi.Api.Runner ReMi.Api** (mandatory)**Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.Ldap** (optional)**Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.Email** (optional)**Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.Jenkins** (optional) **Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.Go** (optional)**Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.Jira** (optional)**Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.Gerrit** (optional)**Install-Package –ProjectName ReMi.Api.Runner ReMi.Plugin.ZenDesk** (optional)

1. The CS-Script package adds a file ‘Scripting.cs’ which can be removed:



1. Open the Global.asax.cs file and change it from inheriting ‘System.Web.HttpApplication’ to inherit from ‘ReMi.Api.WebApiApplication’ instead.



1. Remove or override the Application\_Start method

# Database configuration

ReMi uses MS SQL Server as the data store and will handle all the database objects creation and initialisation. ReMi has not been tested with other data stores; however the architecture does lend itself to implementing other database initialisers.

To configure the database, you need to set the connection string values in the configuration file. Each ReMi plugin is implemented using its own data schema so it is possible for the plugins to share the same database.

By default connection string points to the local ‘remi\_db’ database using Windows Authentication as follows:

<connectionStrings>

<add name="ReleaseManagementConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="LdapConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="EmailMockConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="JenkinsPluginConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="GoPluginConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="JiraConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="GerritPluginConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

<add name="ZenDeskConnection" providerName="System.Data.SqlClient" connectionString="Data Source=localhost;Initial Catalog=remi\_db;Integrated Security=True" />

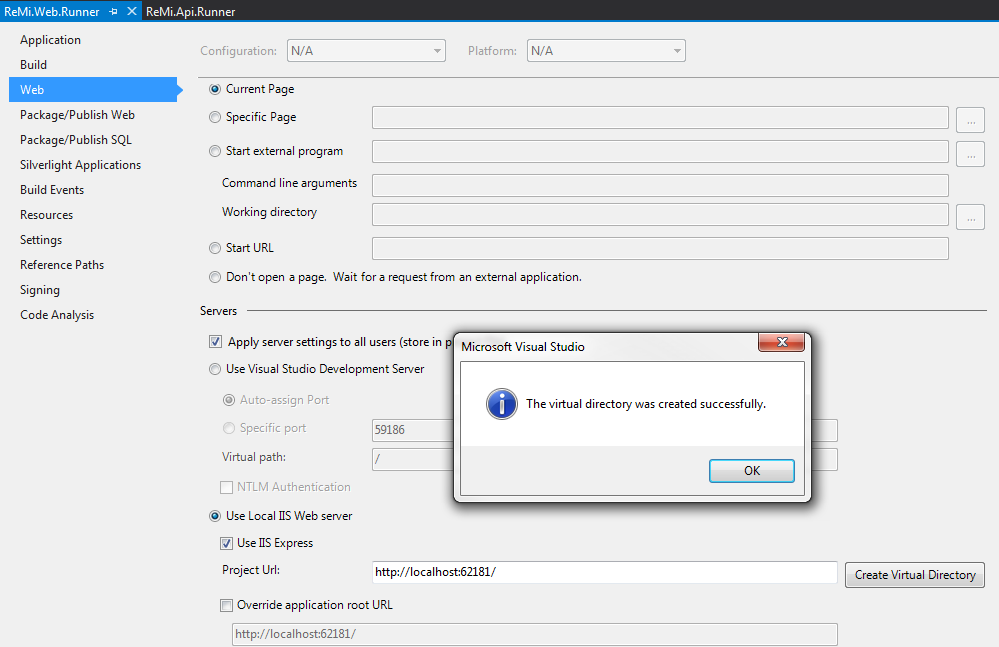
</connectionStrings>

The ‘ReleaseManagementConnection’ setting is ReMi’s main database connection string. Note: if you want ReMi to create the database automatically, the credentials provided require permissions to create databases on the target MS SQL Server.

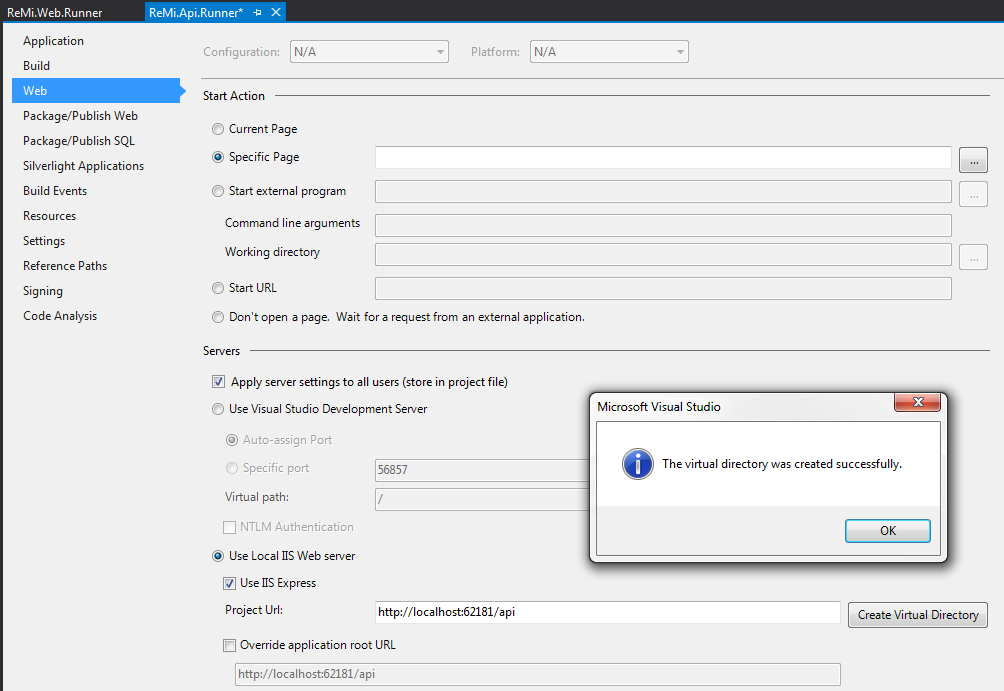
# Configuring IIS Express

To run the application from Visual Studio, the project properties for ReMi.Web.Runner and ReMi.Api.Runner will need to be configured.

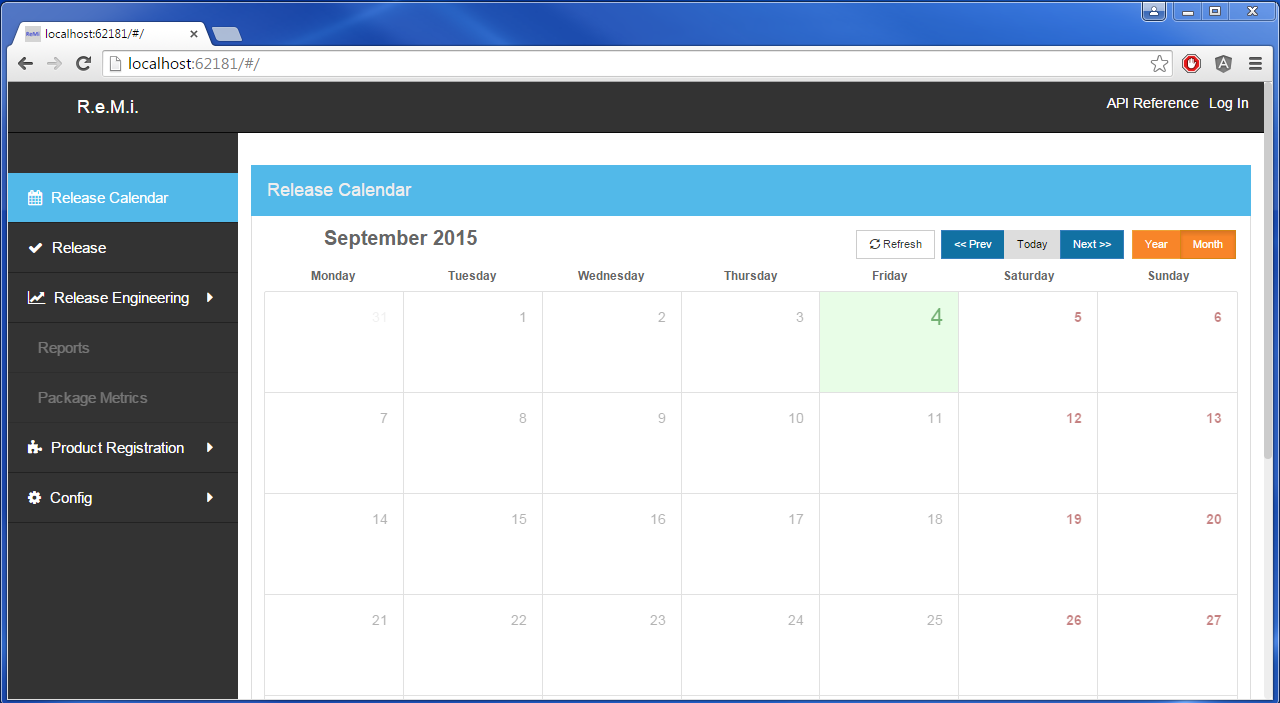
1. In ReMi.Web.Runner properties select the ‘Web’ section, choose *Use Local IIS Web Server*, check *Use IIS Express* and click *Create Virtual Directory*.



1. In ReMi.Api.Runner properties configure the same settings, using the same port number with ‘/api’ appended to the end of the project url path.

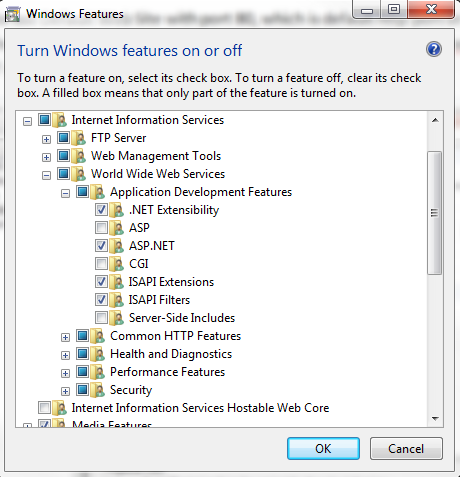


1. Set ReMi.Web.Runner as default project and start the application. If everything is correctly configured ReMi will be loaded into the browser.

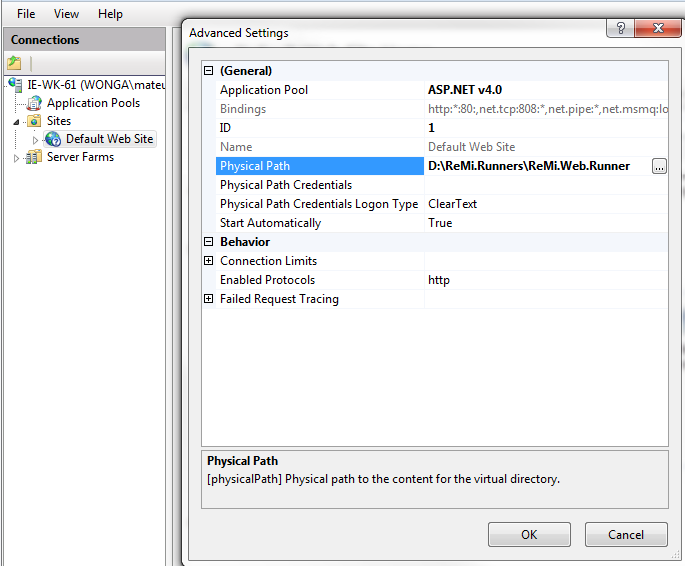


# Configuring ReMi to run under IIS

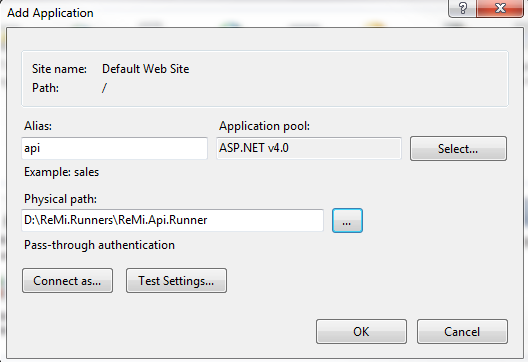
1. Ensure IIS is installed with ASP.NET by checking the installed Windows Features (WinKey + R and type ‘*optionalfeatures’*)



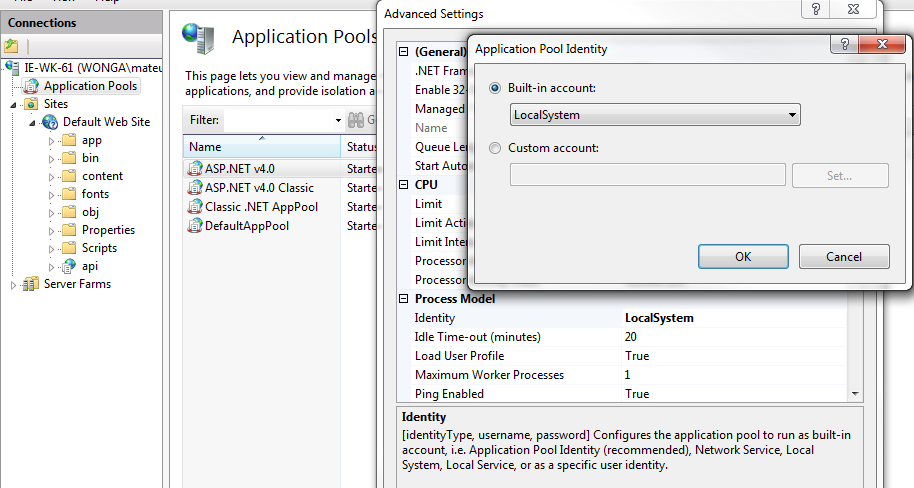
1. Once you have verified that IIS is installed and configured as above, open the IIS configuration manager (WinKey + R and type in ‘*inetmgr’*) to configure the ReMi site.
2. You may configure ReMi to use the Default Web Site with port 80 or you could create a new application using a different port.
   1. Configure the ‘*Physical Path’* of application to point to the ‘ReMi.Web.Runner’ physical path and allocate the application to the ‘ASP.NET v4.0’ app pool.



* 1. In the Web Site created in the step above, add a new application ad set its alias to ‘api’ and allocate the ‘api’ application to the same application pool as the site created in step 3.



* 1. If your local machine has restricted permissions, you may need to change Identity of application pool. To do this, select the app pool, select settings, advanced and set the identity to ‘*LocalSystem’*.

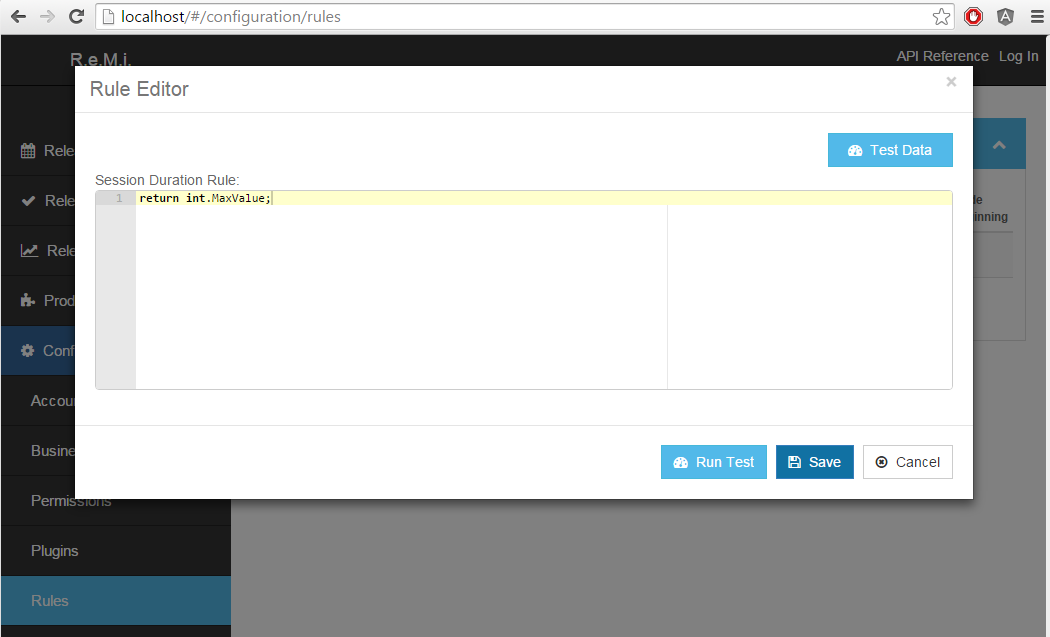


1. With the above configuration you should be able to open ReMi application from a browser using <http://localhost>

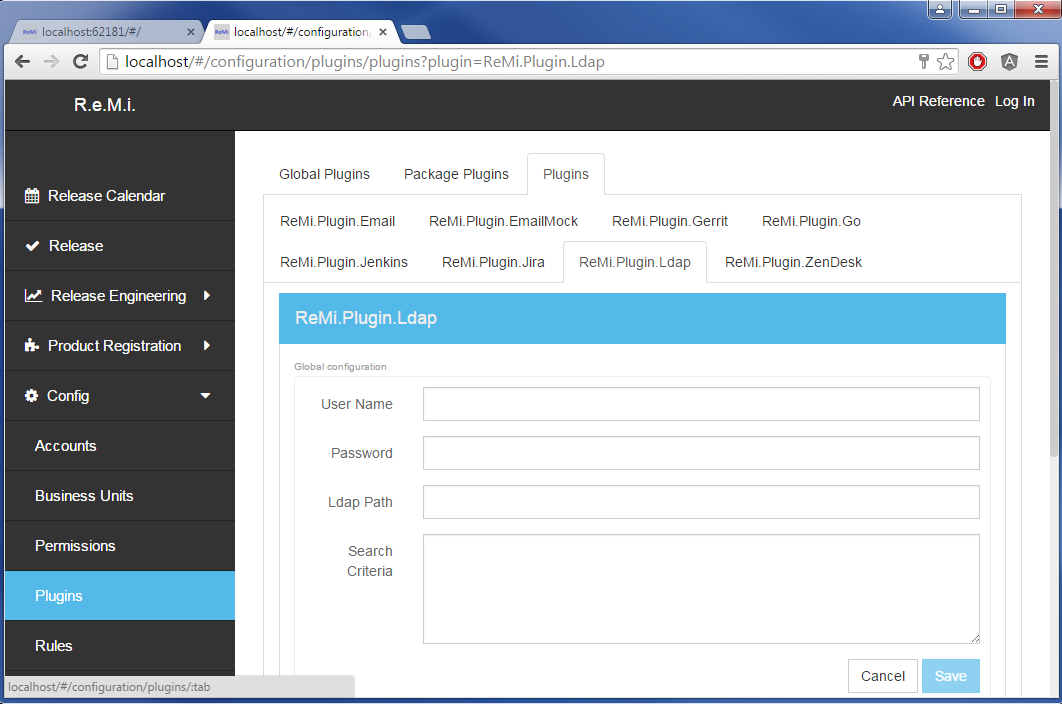
# Basic Configuration

ReMi will not implement any access permissions until the authentication plugin is configured. This allows the first user creating the system to assume the role of Administrator and configure the system as required.

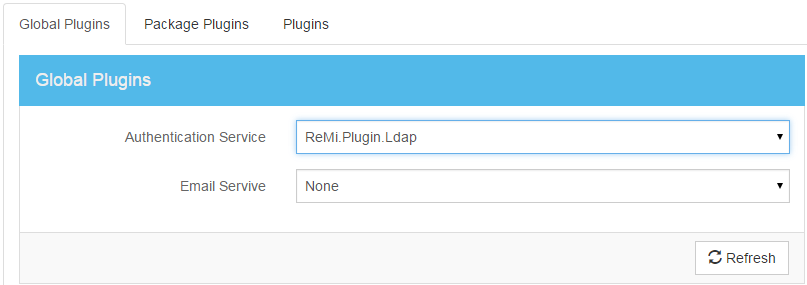
1. Set the user session duration:
   1. Navigate to ‘Rules’ in the ‘Config’ section select ‘SessionDurationRule’ and set it to the desired value. To set it to the maximum allowable value, use ‘return int.MaxValue;’.



* 1. Configure authentication from the Config -> Plugins tab by selecting the Authentication service and add the user name and password for the account that will access Active Directory. Enter the AD search criteria and click save

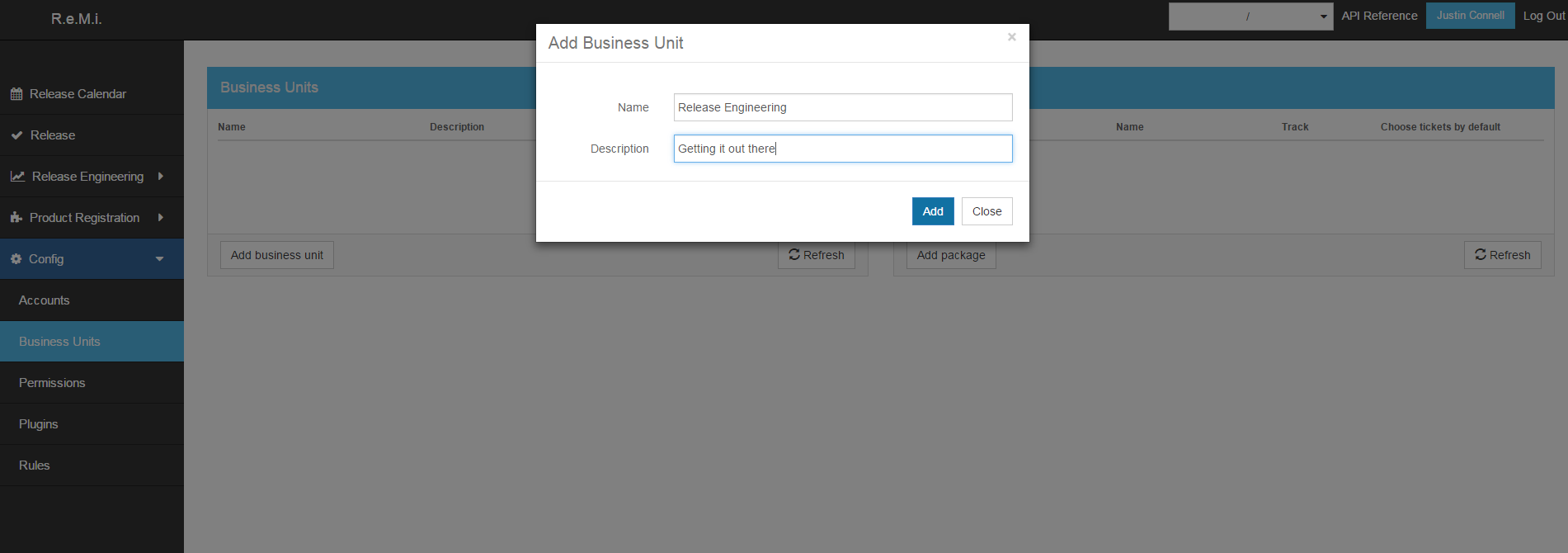


* 1. After configuring the Ldap plugin, you may select it as the default authentication plugin on Global Plugins tab.

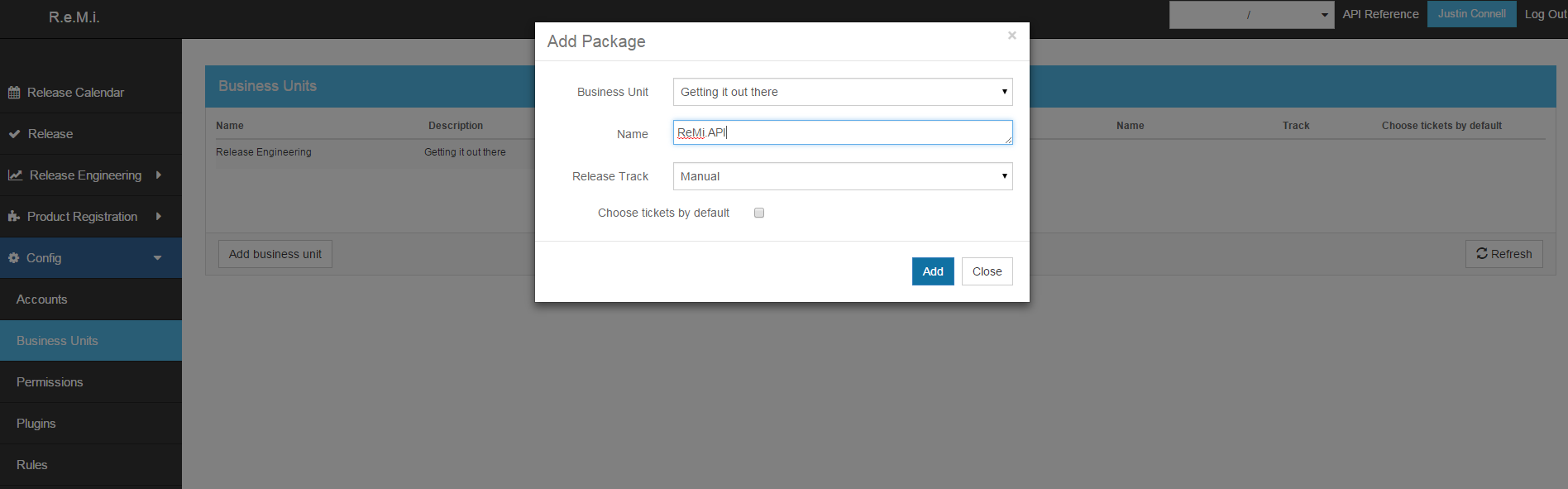


* 1. Test the LDAP configuration is correct by logging in with your Active Directory credentials

1. Next, setup the Business Unit and Package (deployable component).
   1. Navigate to Config -> Business Units and add new business unit. After that add also new package.



* 1. Now add a deployable package



1. Now that we have a business unit with a deployable package configured we can start using ReMi for scheduling and managing releases

This guide introduces the concept of setting ReMi up on a local development environment, the User and Administration Guide will provide more detail on configuring the system for production use.