PeerAMid Developer Documentation

# Overview

PeerAMid is a self-service application for SG&A and Working Capital Analyses. It’s a web application hosted on IIS; users do not need any software other than a browser to access it.

# Setup

## Hosts

We currently have two deployments of PeerAMid, named ‘development’ and ‘production’. Each deployment consists of two virtual machines, one for the database system (SQL Server) and one to host IIS and the PeerAMid code on top of IIS.

Development

Database Server USE1DEVICRSQL01

IIS Server USE1DEVICRWEB02

UAT

Database Server USE1UATICRSQL01

IIS Server USE1UATICRWEB01

Production

Database Server USE1PRDICRSQL01

IIS Server USE1PRDICRWEB02

Each of the database servers is running a version of Microsoft SQL Server on top of Windows Server 2019. The PeerAMid application always connects to the database server with:

Authentication: SQL Server Authentication

User: peeramiduser

Password: a

This user has sufficient access for all our administrative needs, so we use it for that, too.

Each of the IIS hosts is running IIS version 10 on top of Windows Server 2019. There are some details to the configuration of these machines, which is covered in Appendix 1.

An important point is that we (the PeerAMid developers) cannot add (or delete) users of the system. This is done by A&M IT and is briefly described in Appendix 2.

## Developers

Developers working on PeerAMid will need to have Visual Studio installed, at least version 2022. The Community Editions are fine. The developer should install all the C#, ASP, and JavaScript support modules for VS.

The developers should also install Git. The lead developer should have access to the PeerAMid GitHub repository:

GitHub: https://github.com/

User: AlvarezMarsal

Password: votfjrchgunxaebswmqlidpk

Developers will also want to have their W drive mapped to the ‘www’ drive on the development server, [\\USE1DEVICRWEB02\www](file:///\\USE1DEVICRWEB02\www). The ‘Publish’ command in VS will use this when you are deploying to the development IIS server.

# Building the Application

Once a developer’s system is [set up](#_Developers), he can get the source code from GitHub and put it anywhere convenient to him. For the rest of this document, we will assume that he has put it under a directory named C:\AM.

The directory will look like this once the source code is cloned/extracted:

C:\AM (for example)

├— .git

├— Database Snapshot

├— Documentation

├— PeerAMid (the PeerAMid solution folder)

├— .gitattributes

└— .gitignore

The PeerAMid solution folder is named ‘PeerAMid’ and contains two solutions: Support.sln and PeerAMidPortal.sln. Ignore Solution.sln for now; it’s covered under Maintenance, below.

Open PeerAMidPortal.sln with Visual Studio. The Solution Explorer should now show the contents of the solution as:

Solution ‘PeerAMidPortal’

> PeerAMid.Common

> PeerAMid.Core

> PeerAMid.Data

> PeerAMid.DataAccess

> PeerAMidPortal

The first step is to install the NuGet packages that the application needs. Do this by right clicking on the Solution line (the top one) and selecting “Restore NuGet Packages” from the context menu. This will download and install the necessary packages.

Select the ‘DEV’ configuration for development work. For production, switch that to ‘PROD’.

Then right-click the PeerAMidPortal project (the bottom line) and select “Build” from the context menu. The build should not take long and should not produce any warning messages.

# Deploying Updates

Once you’ve built the application, you can deploy it. There are three deployment profiles built into the solution:

Local.pubxml This one deploys the ASP app to the user’s local instance of IIS, for local testing.

DEV-WEB.pubxml This one deploys to the development server.

PRD-WEB.pubxml This one deploys to the production server.

# Notes on the Application

The PeerAMid application works like this:

* Present the user with a page on which the user can choose a company (hereafter called the ‘benchmark’ or ‘target’ company).
* Present the user a page on which the user can choose ‘peer’ companies to compare to the benchmark company.
* Generate a report comparing the benchmark company to those peers by:
  + Gather data on the companies from the PeerAMid database.
  + Write that data to an Excel spreadsheet
  + Call a function exposed by that spreadsheet to process the data and generate a PowerPoint presentation

This is somewhat-serpentine process results in a PowerPoint deck that the user can then customize to serve as a sales tool. The process reflects the history of the application as an evolution of an Excel-based system for generating these reports.

Clearly, it would be better to generate the presentation directly within the application. This transition hasn’t been made due time constraints. This is cause for some concern, since Excel won’t really scale well as many users begin to produce reports simultaneously in the system. The most workable approach would seem to be to migrate all the Excel logic into the C# app and then to use OpenXML (or some similar software) to generate the presentation.

On the plus side, Excel and PowerPoint only need a few moments to produce the presentation. The PeerAMid app includes a ‘grim reaper’ thread that terminates any zombie instances of those apps that might remain afterward.

The user can also run reports against data manually entered by the user; see [*User Private Data*](#_User_Private_Data), below.

# Notes on the Database

Overall, the PeerAMid database is not very complex. It is updated manually a great deal, which has led to some table-to-table relationships being handled informally instead of via defined foreign-key relationships.

The ‘YS’ schema prefix used for many tables is a holdover from the name ‘Yardstick’, which was an earlier version of the project.

## Tables

#### Customers[[1]](#footnote-1), Facts, and the CompanyFactsData view

These Customers and Facts tables contain information for thousands (soon to be tens of thousands) of companies. The tables are linked by a unique identifier for each company (Customers.UID and Facts.UID). The CompanyFactsData view provides a merged view of the two tables along with some resolution of industry names and so forth.

#### Regions and CountryRegions

The world is divided into five regions, roughly by continent. These are enumerated in the Regions table. Countries are assigned to regions in the CountryRegions table, and the Customer table has a Country field assigning each Company to a country. This effectively assigns each Company to a Region. Note that this Company’s relationship with the CountryRegions table is NOT formalized with a foreign-key declaration – it is possible to have a Company with a Country value for which no corresponding record in the CountryRegions table exists. This will cause the Company to be excluded from the CompanyFactsData view, so it will be invisible to the PeerAMid application for most purposes.

#### Sic2D and SicInds

The Sic2D table enumerates several industries and assigns to each a different 2-digit code. The SicInds table enumerates many sub-industries and assigns to each a different 4-digit code. Each Company has both a 2-digit industry code (Customers.Sic2D) and a 4-digit sub-industry code (Customers.Sic\_Code). The first two digits of the Company’s sub-industry code are always the same as the Company’s industry code. All these relationships are informal; there are no constraints or foreign-key relationships that enforce them. It is possible to create a Customer record with a Sic2D and/or Sic\_Code value that doesn’t correspond to anything in the Sic2D or SicInds tables. This will cause the Company to be excluded from the CompanyFactsData view, so it will be invisible to the PeerAMid application for most purposes.

#### Phrases and PhrasesVersion

The Working Capital module of PeerAMid uses ‘phrases’ to generate text inside their reports. (SG&A has text of this sort hard coded into the program). The analyst typically gives the developer a spreadsheet with the phrases and the conditions which trigger them, and the developer updates the database based on that spreadsheet. Ultimately, the phrases are copied into the Excel spreadsheet used for report generation and from thence to the report itself.

#### Application Setting

This table consists of a single record, which records the current fiscal year.

#### Client\_Actual\_Data and UserCompanyMapping

These are described under User Private Data, below.

#### RegisteredUser, UserLoginActivity, and RunAnalysisLog

These three tables provide a user history of sorts. As covered in Appendix 2, the users and their rights are controlled by A&M IT. The RegisteredUser table maintains a list of users that have ever logged in (by their email address). When a new user logs in (one with an email address different from any recorded in the RegisteredUser table), a new RegisteredUser record is generated. The UserLoginActivity table has a record for each user session, recording the user’s ID, the time the session began and the time it ended. The RunAnalysisLog is a bit misnamed, since it records when a report is delivered as a PowerPoint file, rather than when it is run.

# User Private Data

An important feature of PeerAMid was glossed over in the [*Notes on the Application*](#_Notes_on_the) section above: the user can enter company information for his use only. There are two types: user-defined companies, and user-defined cost breakdowns.

User-defined companies are placed in the Customers table, just like the other company data. The difference is that the ‘US1K’ column will have the string ‘Private’ for those companies – all others will have the value ‘Private’.

Both ‘Public’ and ‘Private’ companies can have private cost breakdowns. These are stored in the Client\_Actual\_Data table.

Both types of data are deleted automatically by the system after a configurable number of days. The count of days begins when the user logs out.

# Notes on the Source Code

Text

# Appendix 1: The Web Servers

Any computer that will run the application (i.e., the Development and Production servers, and the developer’s computers) must have a special local user set up: the ComServerExec user.

The PeerAMid ASP application does part of its work by opening and running an Excel spreadsheet and a PowerPoint presentation. This is done via Microsoft’s Interop services, which in turn run on top of the venerable DCOM (Distributed Component Object Model) system, which is an inter-process communication protocol.

The upshot of this is that the PeerAMid app must run by impersonating a user on the IIS server that has the rights to run DCOM, and to access the system desktop (which is required to run Excel and PowerPoint).

Almost all access problems with the application are caused by either (1) changes to the IIS configuration for the application or (2) changes to the access rights of the ComServerExec user. These sometimes seem to happen ‘out of the blue’, caused by some subtle change introduced by the Windows automated update process or by some change IT makes to access rights.

## Setting Up the ComServerExec User

Create a local user named **ComServerExec** on the IIS server. On the Development servers, the password for this user is “@Microsoft”; on the Production servers it is “@P33ram1dus3r”. Note: these values can be modified in the Web.config file, and probably should be. Developers running on their own computers will probably be running the DEV version and should use the same password as the Development server.

Set the user’s properties so that the password never expires.

The ComServerExec needs to be able to start Excel on its own, so do the following:

Run “dcomcnfg.exe”

Navigate to Component Services\Computers\DCOM Config

Find “Microsoft Excel Application” under that node; this will open a properties window.

On the “Identity” tab, select “The interactive user”

On the “Security” tab, and give ALL “Launch and Activation Permissions” to ComServerExec, IUSR and NETWORK SERVICE.

Add the ComServerExec user to the following groups:

Distributed Com Users

Administrators

Remote Desktop Users

IIS\_IUSRS

Several directories will be accessed by the application and by IIS. Grant FULL CONTROL of the following directories to the indicated users and groups:

C:\Windows\SysWOW64\config\systemprofile\Desktop

ComServerExec

IUSR

IIS\_IUSRS

NETWORK SERVICE

C:\www

ComServerExec

IUSR

IIS\_IUSRS

NETWORK SERVICE

**For machines running IIS (not IIS Express):**

Set ComServerExec to automatically log in by editing the Registry. Under the HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon node, add a “DefaultUserName” key with the value “ComServerExec”, a “DefaultPassword” with the correct password as its value, and an “AutoAdminLogin” key with the value 1.

**For machines running IIS Express (not IIS):**

In your source code folder, find a file named “PeerAMid\PeerAMid\PeerAMidPortal\.vs\PeerAMidPortal\config\applicationhost.config”. Locate the authentication section, and make sure that the overrides for anonymous and windows authentication are set to "Allow" in the attributes. This will let you override the authentication in your Web.config file in the application.

<sectionGroup name="authentication">

<section name="anonymousAuthentication" overrideModeDefault="Allow" />

<section name="basicAuthentication" overrideModeDefault="Deny" />

<section name="clientCertificateMappingAuthentication" overrideModeDefault="Deny" />

<section name="digestAuthentication" overrideModeDefault="Deny" />

<section name="iisClientCertificateMappingAuthentication" overrideModeDefault="Deny" />

<section name="windowsAuthentication" overrideModeDefault="Allow" />

</sectionGroup>

Locate the system.webServer section and prevent locks for Anonymous and Windows authentication.

<add name="AnonymousAuthenticationModule" lockItem="false" />

<add name="WindowsAuthenticationModule" lockItem="false" />

## IIS Configuration

Configuring IIS is a messy task, and (as of the date of writing) the program doesn’t provide any way to import and export settings. It must all be done manually. The existing IIS servers are set up correctly. These instructions are for confirming the settings or setting up a new server.

### The Application Pool

In the IIIS PeerAMid Site, open the Application Pool “Advanced Settings” dialog, the settings should be left at the defaults except as follows:

**General**

.NET CLR Version v4.0

**Process Model**

Identity **LocalS**ystem

Idle Time-out (minutes) 60

Load User Profile **False**

## Troubleshooting

Text

# Appendix 2: Users and Rights

The A&M IT department assigns users the rights to use the PeerAMid app. They do this by assigning users to groups:

#### PeerAMidUsersSEC

The user must be a member of this group to access *any* of the PeerAMid functions. Members can access the high level reports.

#### PeerAMidAdmin

Membership in this group lets users access the PeerAMid administrative functions.

#### PeerAMidFeature-SGA

Members of this group can access the full SG&A Diagnostic report.

#### PeerAMidFeature-WC

Members of this group can access the full Working Capital Diagnostic report.

### Blocking Users

Any user can be blocked from access to the PeerAMid system by manually updating their record in the RegisteredUsers database table. If the IsBlocked column value for a user is TRUE, then the user will be blocked at the login screen.

# Appendix 3: Backing Up and Deploying the Database

The method described below is a simple one, resulting in a single text file, a SQL script that re-creates the database from scratch, with all its contents. This wouldn’t be suitable for a larger database but provides the maximum flexibility in our case.

## Backing Up

This process does not have to run on the server machine itself; anywhere that you can reach it with SMSS is fine.

* Start SMSS and connect to the database server.
* Right -click on the ‘PeerAMid’ database and select “Tasks” from the context menu, and then “Generate Scripts…” from the next context menu.
* This will present you with a wizard window for producing various types of scripts. If you see a welcome screen, click Next.
* The wizard should be showing a window with a subtitle “Select the database objects to script.” Choose “Script entire database and all database objects.” Then hit the Next button.
* The next window has an ‘Advanced’ button. Click it.
  + Under **General,** change ‘Check for object existence’ to True.
  + Under **General**, change ‘Script Collation’ to True.
  + Under **General**, change ‘Script DROP and CREATE’ to Script Drop and Create
  + Under General, change ‘Script Logins’ to True
  + Under General, change ‘Types of data to script’ to ‘Schema and data’
  + Under **Table/View Options**, change ‘Script Triggers’ to True.
  + Then click the OK button.
* The wizard offers various options for saving the script. Choose ‘Save as script file’.
* Select ‘Single Script File’.
* Choose a filename for the script. Make sure the name includes the date and time, and either “DEVELOPMENT” or “PRODUCTION”.
* Choose Unicode text.
* Click the Next button.
* The wizard presents a review screen. Check it, then click Next.
* After a short wait, the script will be written.
* Click Finish.
* You’re done. You may want to ZIP the file and place the archive someplace safe.

## Deploying the Database

Large data changes are usually made on the Development server and are later deployed to Production. These are the steps to follow.

* Start SMSS and connect to the Development database server.
* Make a backup of the Development database as described in the previous section.
* Disconnect from the Development Server.
* In SMSS, connect to the Production Server.
* Make a backup of the Production database as described in the previous section.
* Right click on the ‘PeerAMid’ database and select ‘New Query’.
* Open the Development backup script you made earlier in a text editor. Then cut-and-paste it into the query window.
* Click the ‘Execute’ button, then wait while the script runs. Take a nap.

1. The Customers table is a list of Companies, regardless of whether they are customers of A&M or not. Throughout, we use the terms ‘Customer’ and ‘Company’ interchangeably. [↑](#footnote-ref-1)