Profiles Research Networking Software Installation Guide

Documentation Version: December 18, 2025

Software Version: ProfilesRNS_4.0.0

Table of Contents

Introduction

This document describes how to install the Profiles RNS database, website, and APIs. It also provides information about upgrading from previous versions of the software.

If you are upgrading from version 3.1.0 or an earlier version of Profiles RNS, follow the instructions in “Upgrading from a Prior Version”.

If you are performing a new install of Profiles RNS 4.0.0, read the “Hardware and Operating System Requirements” section, and then jump to “Installing the Database” and then continue through the end of this document.

Hardware and Operating System Requirements

Profiles RNS is a Microsoft .NET 4.6.2 website that uses a Microsoft SQL Server 2022 (2019, 2017, 2016) database. Full-Text Search Service must be installed in SQL Server. You can use the same server (or a virtual machine) for the website and database; however, we recommend you separate the two. The database for Profiles is much more resource intense than the website. The database will require about 50 GB for 10,000 people. This is not a lot of space, but having a fast disk and as much CPU and RAM as possible for the database will benefit performance more than building a more robust web server. The website itself uses little disk space and bandwidth. Though, adding CPU and RAM to the web server improves performance of rendering RDF data as HTML.

Download Options

Two download options are provided for Profiles RNS. These are:

1) Release download – Release downloads contain tested versions of the profiles application. The release download contains scripts for creating a new database, as well as upgrade scripts from the previous version. The application is included in both source and binary forms.
    a. Binary - This option is ideal for users who do not have access to Microsoft Visual Studio to compile the code, or users who want the easiest installation option. The binaries are compiled and published in release mode and are ready for use in production systems.
    b. Source - This option is ideal for users who wish to customize their code, and are comfortable compiling the code in Microsoft Visual Studio. Source code should be published in release mode in Microsoft Visual Studio before being used in a production system.
2) GitHub Clone – The latest source code can be cloned from https://github.com/ProfilesRNS/ProfilesRNS. A GitHub clone gives developers access to the latest Profiles RNS development branch. This code may be unstable and should not be used in production systems.

Upgrading from a Prior Version

Profiles RNS includes database scripts to upgrade prior versions of the database to Profiles RNS 4.0.0. This will preserve data, URIs, mapping tables, and ontology extensions. However, all old .NET code (except for the web.config file) will need to be replaced with the latest version. As a result, you will need to reapply any .NET customizations you have made, such as look-and-feel changes, to the new code.

Below are instructions on how to upgrade from Profiles RNS version 3.0.0 onwards. Note that you cannot upgrade directly from older versions, you must follow each set of database upgrade scripts sequentially.

Instructions for upgrading from versions 2.x.x to version 3.0.0 are detailed in the ProfilesRNS_v2.x.x_UpgradeGuide document.

IMPORTANT: Make a backup of your existing Profiles RNS database before applying any of the upgrade steps described below.

Upgrading from Version 3.1.0 to Version 4.0.0

The Database upgrade process from version 3.1.0 to 4.0.0 contains 9 scripts, these must be run in the correct order, or errors will occur.

1) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _CREATE_SCHEMA.sql file in SQL Management Studio, and then click the execute button to run the script.
2) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _ALTER_Tables.sql file in SQL Management Studio, and then click the execute button to run the script.
3) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _CREATE_Tables.sql file in SQL Management Studio, and then click the execute button to run the script.
4) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _ALTER_Functions.sql file in SQL Management Studio, and then click the execute button to run the script.
5) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _CREATE_Functions.sql file in SQL Management Studio, and then click the execute button to run the script.
6) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _ALTER_Procedures.sql file in SQL Management Studio, and then click the execute button to run the script.
7) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Schema _CREATE_Procedures.sql file in SQL Management Studio, and then click the execute button to run the script.
8) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Data.sql file in SQL Management Studio. Click the execute button to run the script. If you have added additional Generic RDF Plugins to profiles, you will need add queries to this script to add a datatype to these. This datatype tells Profiles whether to handle the modules data as

JSON or as text. If you do not plan to upgrade you're the Profiles Application to version 4.0.0 immediately you can ski the first two queries in this script.

9) Open the VersionUpgrade_3.1.0_4.0.0\ProfilesRNS_Upgrade_Account.sql file in SQL Management Studio, and then click the execute button to run the script. This will add permissions for the App_Profiles10 account to execute new stored procedures. If you connect to the database using a different account, change username in this file.

10) The Profiles RNS 4.0.0 code is intended to completely replace any previous version. To replace existing code copy the files other than the web.config from the Website\Binary\Profiles folder to your Profiles virtual directory. Version 4.0.0 contains a complete rewrite of the Profiles UI. If you have made customizations to the UI, or added your own look and feel to Profiles, you will need to reimplement these changes.

11) Add the following keys to your web.config file:

```
<add key="ACTIVITY_LOG_CACHE_EXPIRE" value="180" />
<add key="SEARCH_CACHE_EXPIRE" value="86400" />
<add key="EDITABLE_PAGE_CACHE_EXPIRE" value="3600" />
<add key="GENERATED_PAGE_CACHE_EXPIRE" value="86400" />
<add key="STATIC_PAGE_CACHE_EXPIRE" value="86400" />
<add key="ProfilesRootRelativePath" value="Profiles" />
<add key="ProfilesRootURL" value="https://example.com/profiles" />
```

The default cache expiration values, these set Search results, generated pages (such as networks and concept pages) and static pages to be cached for 24 hours. Editable pages, such as a researcher's profile or a group page to be cached for 1 hour, and the activity log to refresh every three minutes. The values for ProfilesRootRelativePath, and ProfilesRootURL must be set for your environment. The above values assume a Profiles is hosted at a URL of https://example.com/profiles. If instead profiles was hosted at https://example.com (at the root of the server), these values should be "" and https://example.com respectively.

Upgrading from Version 3.0.0 to Version 3.1.0

1) Open the VersionUpgrade_3.0.0_3.1.0\ProfilesRNS_Upgrade_Schema.sql file in SQL Management Studio, and then click the execute button to run the script.

2) Open the VersionUpgrade_3.0.0_3.1.0\ProfilesRNS_Upgrade_Data.sql file in SQL Management Studio. Click the execute button to run the script. Note that this script makes changes to the PresentationXML. If you have previously edited these XML documents, then you may have to reapply those changes.

3) Open the VersionUpgrade_3.0.0_3.1.0\ProfilesRNS_Upgrade_Account.sql file in SQL Management Studio, and then click the execute button to run the script. This will add permissions for the App_Profiles10 account to execute new stored procedures. If you connect to the database using a different account, change username in this file.

4) Delete your existing PubMedDisambiguation_GetPubs job, then open the PubMedDisambiguation_GetPubs.sql file. Modify this file for your configuration by replacing $(YourProfilesDatabaseName) with your database name, and $(YourProfilesServerName) with your database server name. Run

PubMedDisambiguation_GetPubs.sql to create the disambiguation job. Apply your chosen schedule to this job.

5) Delete your existing PubMedDisambiguation_GetPubMEDXML job, then open the PubMedDisambiguation_GetPubMEDXML.sql file. Modify this file for your configuration by replacing $(YourProfilesDatabaseName) with your database name, and $(YourProfilesServerName) with your database server name. Run PubMedDisambiguation_GetPubMEDXML.sql to create the PubMed xml job. Apply your chosen schedule to this job.
6) Delete your existing ExporterDisambiguation_GetFunding job, then open the ExporterDisambiguation_GetFunding.sql file. Modify this file for your configuration by replacing $(YourProfilesDatabaseName) with your database name, and $(YourProfilesServerName) with your database server name. Run ExporterDisambiguation_GetFunding.sql to create the exporter disambiguation job. Apply your chosen schedule to this job.
7) The Profiles RNS 3.1.0 code is intended to completely replace any previous version. To replace existing code copy the files other than the web.config from the Website\Binary\Profiles folder to your Profiles virtual directory. If you have modified your existing version of Profiles RNS, you will need to merge these changes with the source code in Website\SourceCode\Profiles\Profiles and recompile the code.

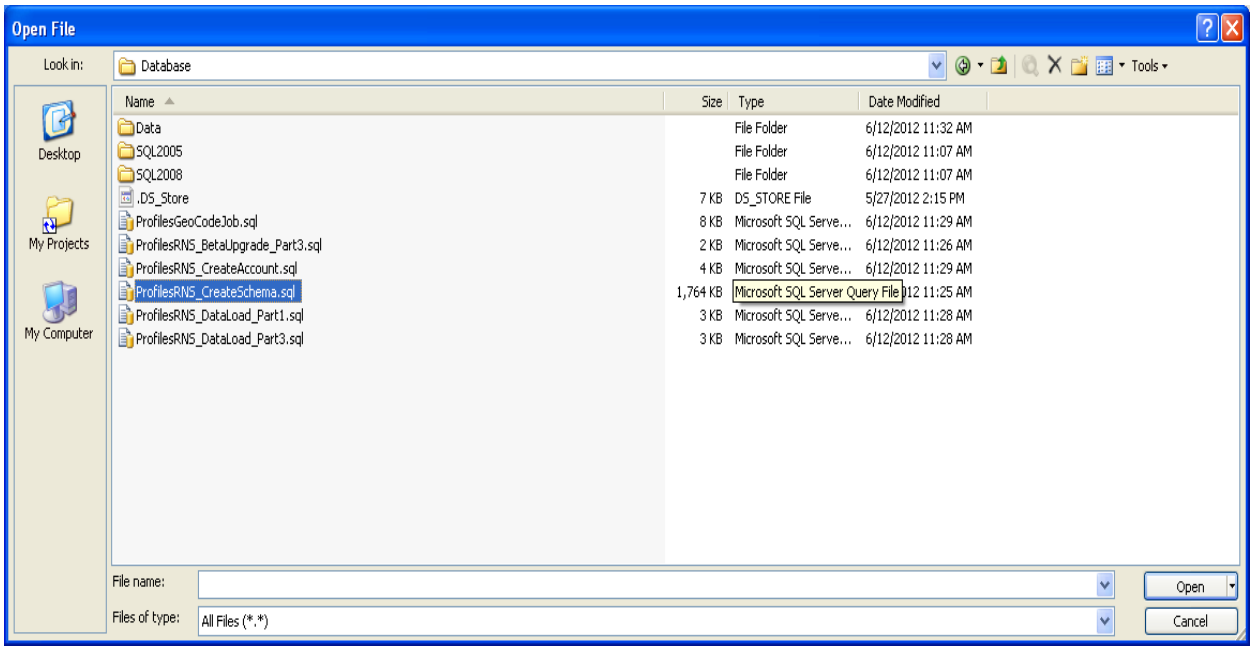
Installing the Database

Follow the instructions in this section only if you are installing a new instance of Profiles RNS. Skip this section if you are upgrading from an existing Profiles RNS database.

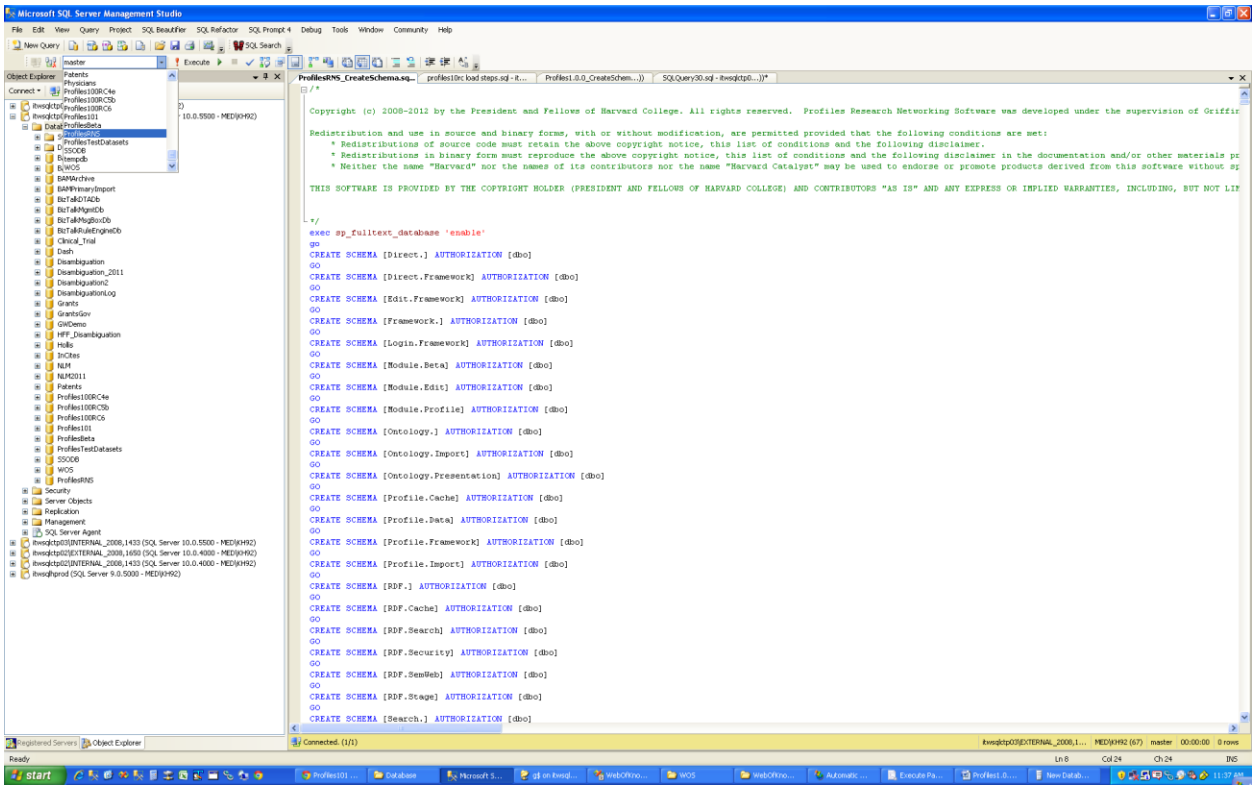
Follow the steps below to create a new database:

1. Create a new empty database.
    a) In SQL Management Studio, open the ProfilesRNS_CreateDatabase.sql file.
    b) The CreateDatabase Script is configured to create a database called ProfilesRNS. No action if needed if the database name is ProfilesRNS. If another name is used, replace all instances of ProfilesRNS with the database name.
    c) Click the execute button to run the script.

2. Add objects tables, stored procedures, and other objects to the new database.
    a) In SQL Management Studio, open the ProfilesRNS_CreateSchema.sql file.



    b) Some of the queries in the create schema script require the database name. These are located in the first section of the script. No action if needed if the database name is ProfilesRNS. If another name is used, edit the first section of the script to replace ProfilesRNS with the database name.
    c) Select the new empty ProfilesRNS database that you created in step 1 from the database drop down list in SQL Management Studio. The script is now pointed to the new database and is ready to execute.



d) Click the execute button to run the script.

3. Create the database user account that will be used by the website.
   a) In SQL Management Studio, open the ProfilesRNS_CreateAccount.sql file.
   b) Edit line 16 to change the default password to a more secure value.
   c) Select the ProfilesRNS database.
   d) Click the execute button to run the script.
4. Import ontology data.
   a) In SQL Management Studio, open the ProfilesRNS_DataLoad_Part1.sql file.
   b) The script contains several steps which are to be executed manually, in sequence. In step 1, it references seven data files (InstallData.xml, VIVO_1.4.owl, PRNS_1.4.owl, SemGroups.xml, SemTypes.xml and MeSH.xml). These files are located in the “Database/Data” folder of the install package. They must be copied to a location on the actual database server, not a separate computer that you are using to connect to the database. Edit the paths to match the location where you placed these files on the database server. [The script will automatically load the data from these files into the database. If you do not have direct access to the database server, you will need to import the data some other way, such as manually by copying the data from the files into SQL Management Studio and creating an INSERT statement to place the data into the proper tables.]. After the files are loaded, the script updates a value in the [Framework].[Parameter] table called basePath. By default, it assumes you are

placing the website at “http://localhost/profiles”. Change this value if needed. Note that basePath should not end with a “/”.

c) Click the execute button to run the script. It might take several minutes to process all the files.

d) In the [Framework.].[Parameter] table, change the value of the parameter RC4EncryptionKey to some secret string. This is used for certain security features in Profiles.

Loading Person Data

Follow the instructions in this section only if you are installing a new instance of Profiles RNS. Skip this section if you are upgrading from an existing Profiles RNS database.

There are five parts to loading person and related data into Profiles RNS: (1) importing SSIS packages into the SQL Server msdb database, (2) importing demographic data, (3) running geocoding, (4) obtaining publications, and (5) running scheduled database jobs. Each part is described below.

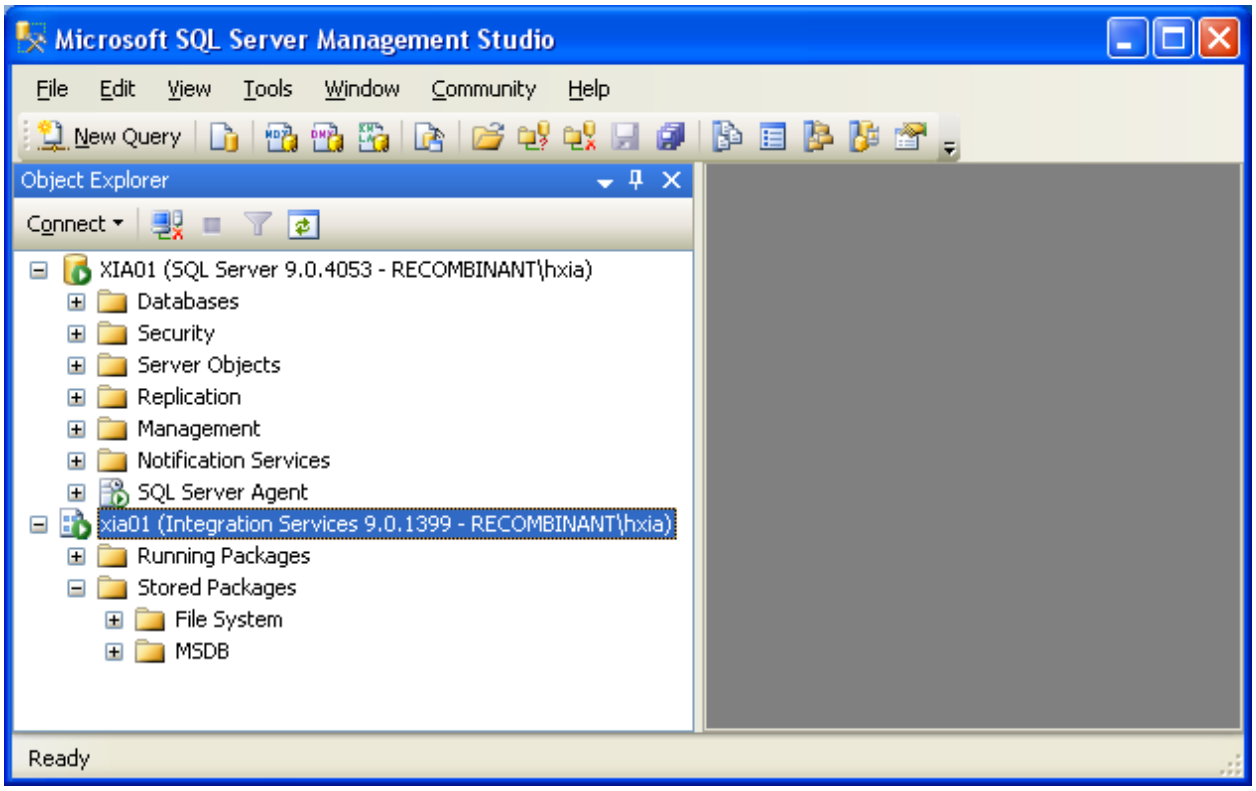
Loading Person Data: Part 1 – Importing SSIS Packages into SQL Server msdb Database

Step 1 below fails on some combinations of SSMS and SQL Server. In this case it is possible to use the command line to install SSIS packages. Run the Import_SSIS_packages.bat file to import SSIS packages from the command line.

We are seeing instances of step 1 succeeding, but the jobs failing to run successfully and returning -2 as the HTTP status. In this case the jobs must be loaded in SQL Server Data Tools (SSDT) , and deployed through SSDT.

1. Before you can import SSIS packages into the SQL Server msdb database, you need to connect to your SQL Server Integration Services from Microsoft SQL Server Management Studio.
    a. Left click **Connect** (the left corner of the Studio)
    b. Pick **Integration Services…**

    You will notice an Integration Services node added to the left panel.



c. Expand newly added **Integration Services** node and you'll see **Running Packages** and **Stored Packages** nodes

d. Expand the **Stored Packages** node and right click **MSDB**, choose **Import Package …,** and then popup an **Import Package** window. From this window, do the following:

   i. **Package location**: **File System**

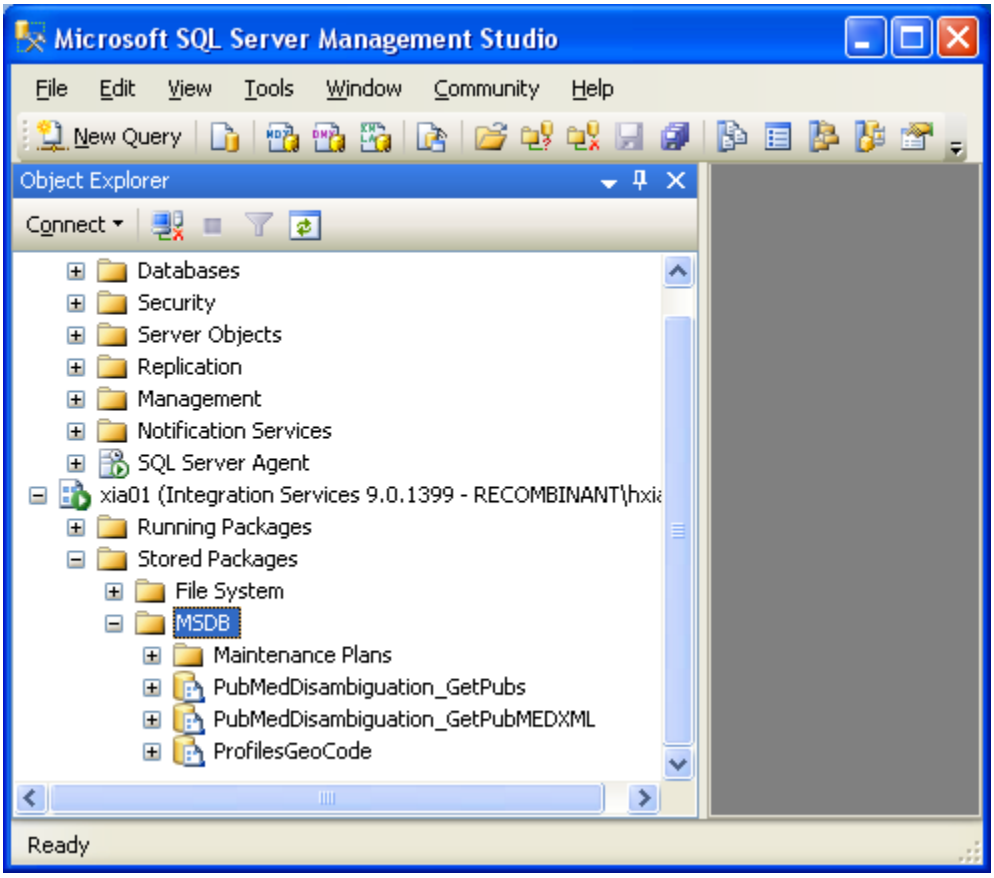
   ii. Navigate and choose a package from your filesystem to install

   iii. Left click **Package name** field and the package name will be filled automatically

   iv. Click **Ok** to install

   You can use this procedure to install the ProfilesRNS_CallPRNSWebservice.dstx package. Note that there are different versions of this package for SQL Server 2012, SQL Server 2014 and SQL Server 2017. These packages are located in the SQL2012, SQL2014, and SQL2017 folders respectfully.

   Successfully installed packages will be displayed under MSDB node as the following:



2. The following scripts that create scheduled jobs need to be modified so that they work in your particular environment:

a. ProfilesRNS_GeoCodeJob.sql
b. PubMedDisambiguation_GetPubs.sql
c. PubMedDisambiguation_GetPubMEDXML.sql
d. ExporterDisambiguation_GetFunding.sql
e. ProfilesRNS_BibliometricsJob.sql

For each of the scripts, modify the following parameters in the sql code:

- Replace `$(YourProfilesServerName)` with the name of your Profiles database server.
- Replace `$(YourProfilesDatabaseName)` with the name of your Profiles database.

3. Execute the scripts you modified in steps #2 and #3 to create the jobs. The following jobs will be created:
   a. ProfilesRNSGeoCode
   b. PubMedDisambiguation_GetPubs
   c. PubMedDisambiguation_GetPubMEDXML
   d. ExporterDisambiguation_GetFunding
   e. ProfilesRNS_GetBibliometrics
4. Update the Profile.Import.PRNSWebservice.Options table to add a google webservices API key.

```
update [Profile.Import].[PRNSWebservice.Options] set apikey = '<your google api key here>' where job = 'geocode'
```

To get an API key follow the instructions here: https://developers.google.com/maps/documentation/geocoding/get-api-key. Note that you should set use a different key for geocoding and for displaying maps. Your display key will be publicly visible, and should be restricted such that it is not valid for other API calls.

Loading Person Data: Part 2 – Importing Demographic Data

Profiles requires that you provide basic demographic data about people. The general process is that you place the data in a set of "import" tables, and then Profiles will copy the data into the actual tables used by the website. During this step, Profiles can automatically create unique IDs for people and generate several lookup tables. There are several concepts to be aware of with how Profiles handles person data:

1. Profiles makes a distinction between the people who have profiles (Persons) and the people who can login to the website (Users). In general, Persons will be a subset of Users. At a typical academic institution, the Persons will be faculty, and the Users will be the faculty, staff, and students. Note that in order for someone to be able to use features of the site that require a login, such as "active networking" and "proxies", he or she will need to have a user account.
2. There are four main import tables:
   a. The [Profile.Import].[Person] table has one row per person and includes fields such as first name, middle name, last name, name suffix, email, phone, fax, and address.

b. The [Profile.Import].[PersonAffiliation] table lists a person's titles, institutions, departments, divisions, and faculty rank (e.g., "associate professor"). This table can have multiple rows per person, reflecting the multiple jobs or roles a person has in your organization.

c. The [Profile.Import].[PersonFilterFlag] table allows you to extend the data model with custom Boolean flags that are relevant to your organization, such as "emeritus", "visiting", "student", etc. There can be multiple flags per person. Flags can be grouped into categories. For example, "faculty", "staff", and "student" can be grouped into a category named "job type".

d. The [Profile.Import].[User] table has one row per user and includes basic user name and affiliation information. The [Profile.Import].[User] table is used to create accounts for individuals who need access to the website, but will not have their own profiles. The same individual should not be listed in both the [Profile.Import].[Person] and [Profile.Import].[User] tables.

3. The raw HR data from many institutions needs to be modified before Profiles can copy it from the import tables into the actual tables used by the website. To assist with this process, we made the column sizes in the import tables longer than the maximum allowed length in the actual tables, and in some cases we made columns in the import table of type nvarchar when they are numeric in the actual tables. This will reduce errors when inserting the raw HR data into the import tables, but you must perform your own validation and cleanup to make sure the final data in the import tables meet the size and type limits as outlined in the column definitions below.

4. There is an "all-or-nothing" approach in the import tables with respect to nulls in optional columns. If you do not want to use an optional column, then all values in it must be null. If you want to use an optional column, there cannot be any nulls in it—use an empty string instead.

5. Each of the import tables has a field named "internalusername". This should be some unique value that you use for each person and user that you load into Profiles. The internalusername allows Profiles to join the import tables during the data load process. You should always use the same internalusername for a given person or user each time you load that individual into Profiles. The internalusername is not displayed on the Profiles website. Instead, for each internalusername, Profiles will create either a PersonID or a UserID, and that value will be displayed on the website. During the load process, you can indicate that you want the PersonID and UserID to be equal to the value of the internalusername; otherwise, Profiles will create its own values based on sequential integers.

6. Do not make changes to the data in the actual tables used by Profiles. Instead, always place corrections or updates in the import tables, and re-run the import scripts. Data is copied to multiple tables within Profiles to improve performance; and, if you change it in one place and not the others, it can result in foreign key violations or cause the website to crash.

7. The Profiles import process does not fully validate the data in the import tables before copying it to the tables actually used by the website. This is a known limitation of the software. If the import process is run with invalid data in the import tables, you might need to restart from scratch with the original database that came with the software.

Below are the column definitions for each of the import tables. The Data Type and Load Length describe the columns in the import tables. The Max Length corresponds to the columns into

which the data is copied during the import process. Despite the size of the columns in the import tables, if their length exceeds the Max Length, then the import process might fail. Also, note that some columns, such as [Profile.Import].[Person].floor are type nvarchar in the import table, but need to have numeric values for the import process to work.

1. Table: [Profile.Import].[Person]

Column	Data Type	Load Length	Max Length	Category
Internalusername	nvarchar	2000	50	Required
Firstname	nvarchar	2000	50	Optional
Middlename	nvarchar	2000	50	Optional
Lastname	nvarchar	2000	50	Required
Displayname	nvarchar	2000	255	Required
Suffix	nvarchar	2000	50	Optional
addressline1	nvarchar	2000	55	Optional
addressline2	nvarchar	2000	55	Optional
addressline3	nvarchar	2000	55	Optional
addressline4	nvarchar	2000	55	Optional
Addressstring	nvarchar	2000	1000	Required
State	varchar	1000	2	Optional
City	varchar	1000	100	Optional
Zip	varchar	1000	10	Optional
Building	nvarchar	2000	255	Optional
Room	nvarchar	2000	255	Optional
Floor	nvarchar	200	int	Optional
Latitude	float		decimal	Optional
Longitude	float		decimal	Optional
Phone	nvarchar	2000	35	Optional
Fax	nvarchar	2000	25	Optional
Emailaddr	nvarchar	2000	255	Optional
Isactive	bit		bit	Required
Isvisible	bit		bit	Required

Notes:

a. The PubMed disambiguation process uses the firstname, middlename, lastname, suffix, and emailaddr columns. Therefore, although only lastname is required, providing values for the other columns will greatly aid disambiguation.

b. The columns addressline1, addressline2, addressline3, and addressline4 are used by the website to display a person's address. The addressstring, state, city, and zip columns are not displayed on the website.

c. The addressstring column is used during the geocoding process to determine the latitude and longitude of the person. The addressstring should be a valid street address (i.e. street number, city, state, zip) and should not contain department names, room numbers, mailbox numbers, etc. The addressstring column is only required if you want to be able to display the location of people on a map or take advantage of physical distance metrics in Profiles. Otherwise, you can leave it blank. Note that the addressstring column is not automatically formed from the addressline1-4 columns and vice versa; in general, you will want to list the address in both places so that it appears on the website AND the person appears on maps. The latitude and longitude columns will override the results of the automatic geocoding, which can be useful if you do not have a precise street address for a person.

d. The values of the state, city, and zip columns are included in the RDF representation of a person, but they are not displayed on the website.

e. The building, room, and floor columns are not displayed on the website. They are only used to estimate the physical distance between people who share the same street address.

f. If isactive=1, then a profile will be created for the person. If isactive=0, then the profile will be removed from the website. Note that changing isactive=0 will not deactivate the person's corresponding user account, and the person will still be able to login to Profiles. To deactivate a user account, manually change this person's record in the user (not [Profile.Import].[User]) table to isactive=0.

g. If isvisible=1, then the content of a profile will be displayed when a user goes to its URL. If isvisible=0, then the profile will be replaced by a message that states that it is not available at this time. However, if isvisible=0, then that person will still be listed in other people's networks and in search results.

2. Table: [Profile.Import].[PersonAffiliation]

Column	Data Type	Load Length	Max Length	Category
internalusername	nvarchar	2000	50	Required
title	nvarchar	2000	200	Optional
emailaddr	nvarchar	2000	200	Don't Use
primaryaffiliation	bit		bit	Required
affiliationorder	tinyint		int	Required
institutionname	nvarchar	2000	500	Optional
institutionabbreviation	nvarchar	2000	50	Optional
departmentname	nvarchar	2000	500	Optional
departmentvisible	bit		bit	Optional
divisionname	nvarchar	2000	500	Optional
facultyrank	varchar	1000	100	Optional
facultyrankorder	tinyint		tinyint	Optional

Notes:

a. Each person must have exactly one row in [Profile.Import].[PersonAffiliation] where primaryaffiliation=1. For all additional affiliations, set primaryaffiliation=0.

b. The affiliationorder for a person's primary affiliation (primaryaffiliation=1) should be set to 1. All other affiliations for a person should be sequentially ordered (e.g., affiliationorder=2, affiliationorder=3, etc.). The same person should not have two affiliations with the same affiliationorder value.

c. Departmentvisible is required if using department names. Set departmentvisible=1 if you want the corresponding departmentname to appear in the Department drop-down menu on the Profiles Search form. Otherwise, set departmentvisible=0.

d. The institutionabbreviation is not displayed on the website, but it is used during the data load process. There must be a one-to-one mapping between institutionname and institutionabbreviation. We suggest setting these two columns to the same value if possible.

e. The emailaddr column is not used by Profiles; however, you must set these columns to NULL for the import process to work properly.

f. Facultyrankorder is required if you are using the facultyrank column. Every distinct facultyrank value in the [Profile.Import].[PersonAffiliation] table needs to have a different facultyrankorder. (Unlike affiliationorder, which is by person, the facultyrankorder is global for the table.) For example, if the faculty ranks in your institution are Professor, Associate, and Assistant, then the facultyrankorder should be 1 for every affiliation whose rank is Professor, 2 for every affiliation whose rank is Associate, and 3 for every affiliation whose rank is Assistant. Note that a person might have two affiliations with the same facultyrank, in which case both affiliations will also have the same facultyrankorder.

3. Table: [Profile.Import].[PersonFilterFlag]

Column	Data Type	Load Length	Max Length	Category
Internalusername	varchar	50	50	Required
Personfilter	varchar	50	50	Required

Notes:

a. After running the data load process, the [Profile.Data].[Person.Filter] table will be populated with a distinct list of personfilter values from the [Profile.Import].[PersonFilterFlag] table. The [Profile.Data].[Person.Filter].PersonFilterCategory and [Profile.Data].[Person.Filter].PersonFilterSort columns will be set to NULL; however, you must manually enter values into these columns for the person filters to appear on the website. Person filters with the same PersonFilterCategory will be grouped under the same heading in the Profiles Search form drop-down menu. The PersonFilterSort column is used to order the person filters in the Profiles Search form drop-down menu.

b. The PersonFilter and PersonFilterCategory values will be specific to your institution. For example, PersonFilters "faculty", "staff", and "student" can be grouped into a PersonFilterCategory named "job type"; "clinical" and "research" can be grouped into "faculty type"; and "past projects" and "current opportunities" can be grouped into "mentoring".

4. Table: [Profile.Import].[User]

Column	Data Type	Load Length	Max Length	Category
Internalusername	nvarchar	2000	50	Required
Firstname	nvarchar	2000	100	Optional
Lastname	nvarchar	2000	100	Required
Displayname	nvarchar	2000	255	Required
Institution	nvarchar	2000	500	Optional
Department	nvarchar	2000	500	Optional
Canbeproxy	bit		bit	Required

Notes:

a. Only list individuals in this table who are not listed in the [Profile.Import].[Person] table.

b. Set canbeproxy=1 if the user is allowed to be an editing proxy for another person with a profile. Otherwise, set canbeproxy=0.

There are several mistakes that users frequently make when entering data into the import tables. Double check that you have not done any of these common errors:

1. ERROR: There are values in the import tables that are longer than the allowed Max Length.
2. ERROR: The same internalusername value is being used more than once in the [Profile.Import].[Person] or [Profile.Import].[User] tables.
3. ERROR: There are nulls in a column that also has non-null values.
4. ERROR: The columns isactive and isvisible are set to 0 when you intended for that person to be shown on the website.
5. ERROR: The column addresslineN is being used, but addressstring is null or empty (or vice versa).
6. ERROR: The required column [Profile.Import].[PersonAffiliation].primaryaffilation is set to NULL.
7. ERROR: There is more than one record per person in [Profile.Import].[PersonAffiliation] with primaryaffilation=1.
8. ERROR: A person does not have any records in [Profile.Import].[PersonAffiliation] with primaryaffiliation=1.
9. ERROR: The required column [Profile.Import].[PersonAffiliation].affiliationorder is set to NULL.
10. ERROR: The same [Profile.Import].[PersonAffiliation].affiliationorder is being used more than once for the same person.
11. ERROR: The same facultyrankorder is being used for two different facultyranks, or two different facultyrankorders are being used for the same facultyrank.

Once you have placed data into the import tables, confirm that you have not made any of the above errors by executing the stored procedure "**[Profile.Import].ValidateProfilesImportTables**". It will generate a report listing any data problems that it finds. Note that this procedure does not check for all possible errors. It only searches for the most common problems.

Once you have validated your data, execute the stored procedure "**[Profile.Import].LoadProfilesData**". This will parse the data in the import tables and populate the actual person, user, and related tables. This procedure takes 1 input parameter, @use_internalusername_as_pkey. If this is set to 1, then Profiles will use the internalusername columns in the [Profile.Import].[Person] and [Profile.Import].[User] tables as the PersonID and UserID. Otherwise, Profiles will generate its own unique values using sequential integers.

Loading Person Data: Part 3 – Geocoding

Troubleshooting: If geocoding fails, and returns a http status code of -2 this means that the compiled SSIS package is not compatible with the SQL Server installation. In this case, delete the ProfilesRNS_CallPRNSWebservice SSIS package, load the ProfilesRNS_CallPRNSWebservice.dstl file in SQL Server Data Tools (SSDT), and deploy the package to SSIS through SSDT.

Profiles uses the Google geocoding API to convert addresses in the addressstring column of the [Profile.Import].Person table to latitude and longitude coordinates. Google's geocoding API is a paid for service, we anticipate that most Profiles institutions will not exceed the free monthly quota, however billing must be configured, and an API key generated prior to running geocoding. Keys can be created on Google Cloud Platform console https://console.cloud.google.com .

You will need to create two keys for profiles. One for geocoding, this key should be restricted to the Geocoding API, and may also be restricted by IP address. The second key is for displaying maps on the site. Maps are displayed using client side javascript which makes the key public. To prevent your key being misused you must restrict this key to HTTP Referrers, add a website restriction to your profiles URL and restrict the key to only work on the Maps Javascript API. You can also set quotas to limit use of the key.

Add your Geocoding API key to the PRNSWebservice.Options table by running the following query:

```
update [Profile.Import].[PRNSWebservice.Options] set apiKey = '<google key here>' where
job = 'geocode'
```

In the SQL Agent folder in SQL Server Management Studio, expand the Jobs folder, right-click the **ProfilesRNSGeoCode** job and choose "Start at Step…". This will send each unique addressstring in the database to the Google web service API, which will return the coordinates. Note that Profiles can only process 3 unique addresses per second because of Google's policy on how frequently its API can be called.

Loading Person Data: Part 4 – Obtaining Publications

Troubleshooting: If obtaining publications fails, and returns a http status code of -2 this means that the compiled SSIS package is not compatible with the SQL Server installation. In this case, delete the ProfilesRNS_CallPRNSWebservice SSIS package, load the ProfilesRNS_CallPRNSWebservice.dstl file in SQL Server Data Tools (SSDT), and deploy the package to SSIS through SSDT.

In order for Profiles to automatically locate publications for people, you first need to provide a list of affiliation strings in the [Profile.Data].[Publication.PubMed.DisambiguationAffiliation] table. These are phrases, which can include wildcard characters ("%"), that represent the most likely ways that your researchers will list their affiliations in Medline/Pubmed. Strings are not case sensitive. Selecting affiliation strings is somewhat of an art. The more precise the strings, the

easier it is for Profiles to find publications. However, if the strings are too narrow in scope, Profiles might miss some articles. Examples of strings that we use at Harvard include:

```
%Harvard Medical School%
%Beth Israel Deaconess Medical Center%
%BIDMC%
%@hms.harvard.edu%
%Children's Hospital%02115%
```

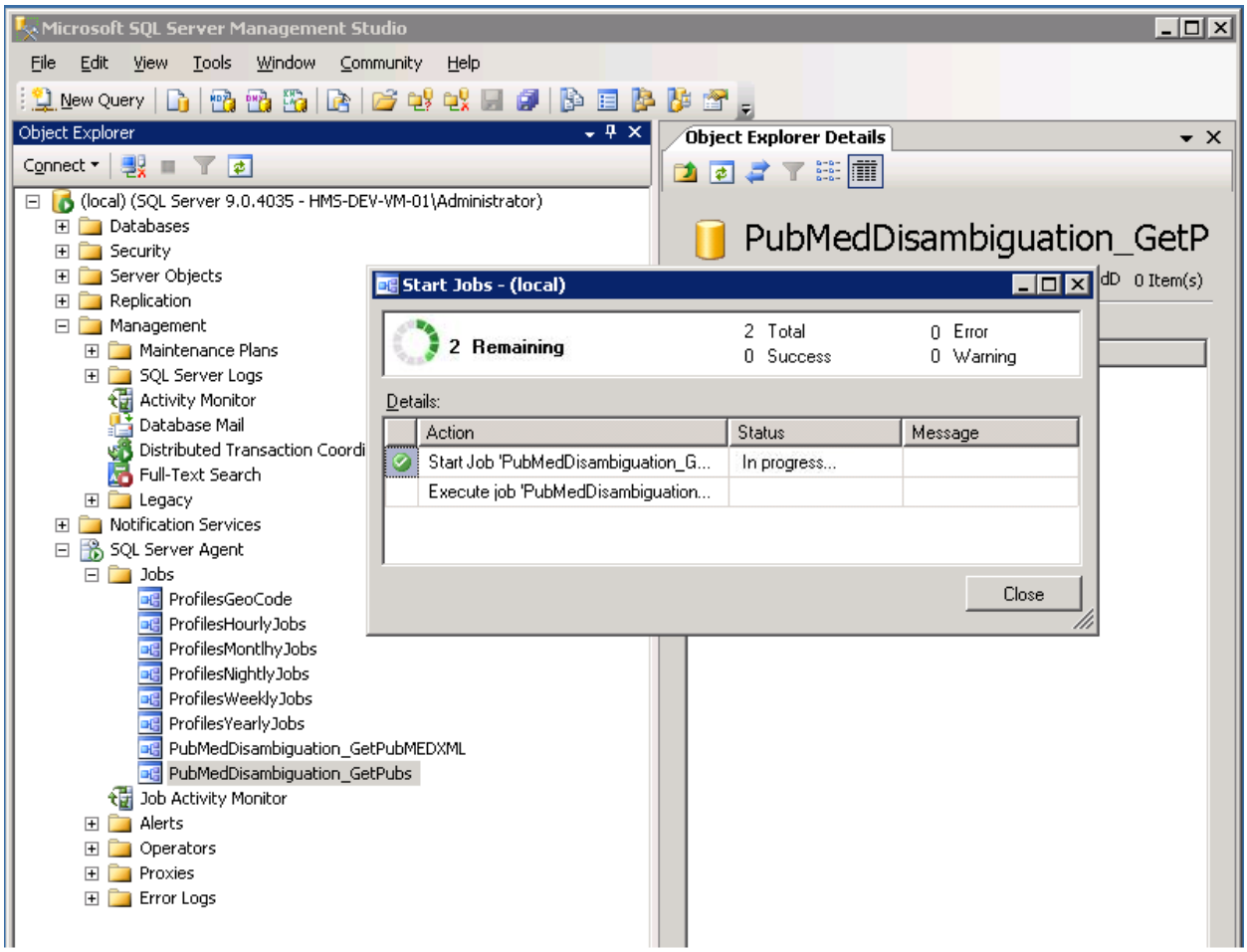
Example:

```
insert into [Profile.Data].[Publication.PubMed.DisambiguationAffiliation]
   (affiliation) values ('%Harvard Medical School%')
insert into [Profile.Data].[Publication.PubMed.DisambiguationAffiliation]
   (affiliation) values ('%Beth Israel Deaconess Medical Center%')
insert into [Profile.Data].[Publication.PubMed.DisambiguationAffiliation]
   (affiliation) values ('%BIDMC%')
insert into [Profile.Data].[Publication.PubMed.DisambiguationAffiliation]
   (affiliation) values ('%@hms.harvard.edu%')
insert into [Profile.Data].[Publication.PubMed.DisambiguationAffiliation]
   (affiliation) values ('%Children's Hospital%02115%')
```

Examples of strings that we do not use at Harvard because they would be too broad are:

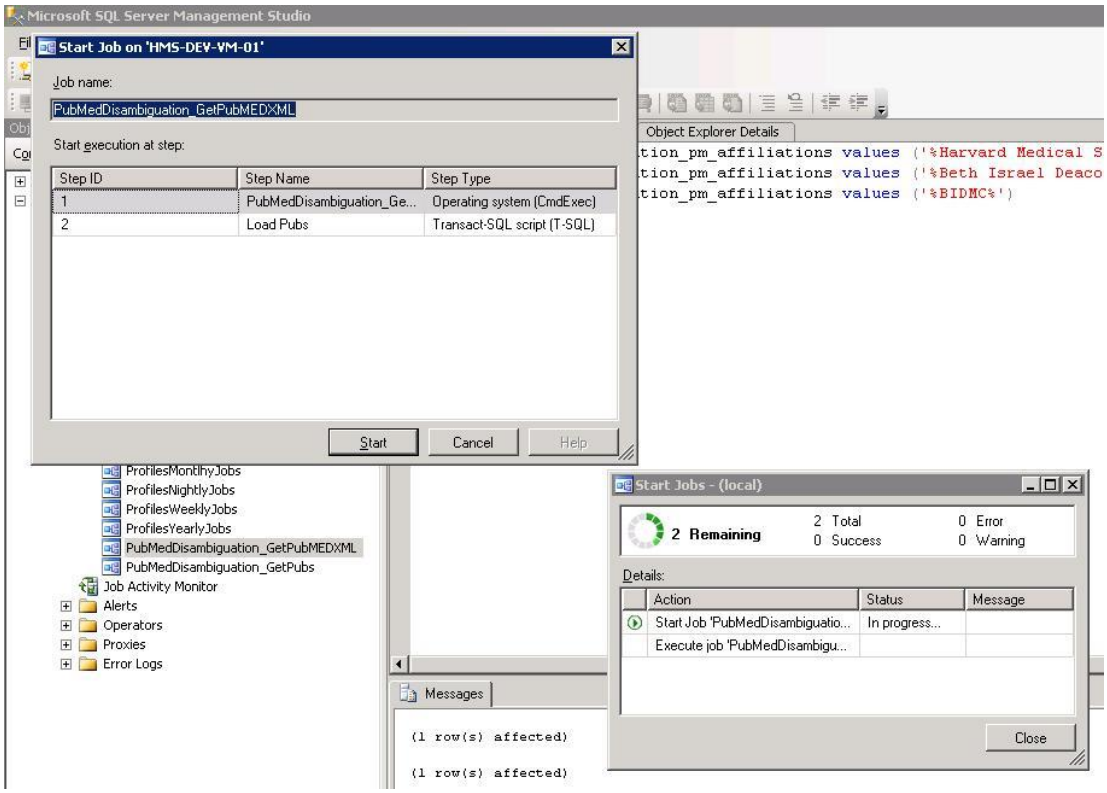
```
%Beth Israel%
%Department of Medicine%
```

Once the person data is loaded, and you have entered your affiliation strings, run the **PubMedDisambiguation_GetPubs** job to call the Profiles Disambiguation Engine web service to find Medline/Pubmed articles for people.



Once this job has completed (typically several hours), you should run the **PubMedDisambiguation_GetPubMEDXML** job. This job will retrieve the full xml for the pubmed articles and parse it in your local profiles instance.

Once this job has completed (typically several hours), you should run the **ProfilesRNS_BibliometricsJob** job. This job will retrieve citation counts and other bibliometric data used by profiles.



There are a few important technical notes about the service:

1. The service will take about 5 seconds per person on average, provided you are the only one using the service. If another institution is calling the service at the same time, the run time will be slower.
2. It is possible to host your own local instance of this service. However, its hardware and storage requirements are significantly greater than the main Profiles database and website. For example, you will need to have a local copy of the entire Medline database, which is several hundred gigabytes.

Below are a few general comments about the disambiguation engine:

1. Although the affiliation strings help the service find publications, it does not limit the search. The affiliation strings are used to identify “seed” publications. These are publications that are most likely correct matches. The disambiguation engine then searches all of Medline/Pubmed, using information about the seed publications, such as their titles, MeSH terms, coauthors, and journals, to find additional articles.
2. All publications are assigned a match probability. By default, the disambiguation engine uses a 98% probability threshold, meaning it will only return publications that are very likely correct matches. You have the option of lowering this threshold. This will reduce the chances that correct publications are missed, but it will increase the chances that incorrect publications are added to people’s profiles. In general, select a low threshold if your goal is to create the “most accurate” profiles, meaning as many people as possible have close to correct publication lists. However, select a high threshold if your goal is to create the “cleanest” search results and passive networks. We set the default threshold high because it is easy for faculty (or their proxies) to add missing publications, but the website loses much of its value if the search results return the wrong people or if passive

networks (e.g., top keywords, co-authors, similar people, etc.) contain meaningless information. Note that just a single incorrect publication can greatly alter a person's passive networks, but even multiple missing publications will have far less effect because an expert in a field will have many other publications in that same area. To change the threshold, modify the @threshold variable value defined within the [Profile.Data].[Publication.PubMed.GetPersonInfoForDisambiguation] stored procedure.

3. Profiles will have the most difficulty with common names (e.g., J Smith), names with multiple parts (e.g., a hyphenated last name), names with foreign characters, and people who only recently joined your organization. We are continually working to improve the disambiguation engine to address these issues.
4. If two or more people in your Profiles database share the same first name and last name, then this will lower the publication match probabilities for those people. This logic is defined in the [Profile.Data].[Publication.PubMed.GetPersonInfoForDisambiguation] stored procedure when it calculates the value for the XML tag "LocalDuplicateNames".
5. The disambiguation process includes an optional parameter, "RequireFirstName", which when set to true, will only find seed publications where the author's entire first name (not just the initial) is used. If two or more people in your Profiles database share the same last name and same first name initial, then this parameter is set to true. There are other use cases when you might want to use this option. For example, young investigators (e.g., post-docs) have few publications before 2002, the year when Medline began including author first names. By requiring a first name match for these people, it should have little effect on correct publication matches, but it has the potential to eliminate older publications that might be incorrect matches. To add this or other custom logic to control the RequireFirstName paramenter, modify the code in the [Profile.Data].[Publication.PubMed.GetPersonInfoForDisambiguation] stored procedure.
6. Users or their proxies can manually edit their publication lists within Profiles. The disambiguation engine uses these modifications to improve its search. For example, if a publication was manually added, it will never be automatically removed. If a publication was manually deleted, it will never be automatically re-added. Manually added publications are used as additional seed publications for subsequent calls to the disambiguation engine. In other words, if users need to make corrections to their publication lists, Profiles will learn from this, and it will become more likely that future corrections will not be necessary.

Loading Person Data: Part 5 – Obtaining Funding Data

The [Profile.Data].[Funding.DisambiguationOrganizationMapping] table maps the institutions in your Profiles database to organization names used by the NIH. You can set the institution ID in this table to null to map all institutions in your database to an NIH organization name. The organization names have to be exactly as they appear in the NIH RePORTER website. They cannot contain wildcard characters. If you do not know the exact organization names, go to the search form on the NIH RePORTER website:

https://projectreporter.nih.gov/reporter.cfm

Enter at least three characters into the "Organization" form field and then click the "Lookup" button. A list of matching organization names will be shown. Note that some organization names will be followed by a comma and then the city, state, or country name. You should only use the organization name, without the location part, in the Profiles

[Profile.Data].[Funding.DisambiguationOrganizationMapping] table. For example, a search for "Harvard" returns "Harvard University, Cambridge, MA", but organization name is only "Harvard University".

The example below maps all people to Harvard University, but limits schools and affiliated hospitals to particular institutions. Note that the mappings are many-to-many, which means multiple institutions can be mapped to the same organization name, and an institution can be mapped to multiple organization names.

Example:

```
insert into [Profile.Data].[Funding.DisambiguationOrganizationMapping]
   (InstitutionID, Organization) values (NULL, 'HARVARD UNIVERSITY')

insert into [Profile.Data].[Funding.DisambiguationOrganizationMapping]
   select InstitutionID, 'HARVARD SCHOOL OF PUBLIC HEALTH' from
   [Profile.Data].[Organization.Institution] where InstitutionName =
   'Harvard School of Public Health'

insert into [Profile.Data].[Funding.DisambiguationOrganizationMapping]
   select InstitutionID, 'CHILDREN''S HOSPITAL CORPORATION' from
   [Profile.Data].[Organization.Institution] where InstitutionName =
   'Children''s Hospital Boston'
```

Once the organization names are loaded, run the **ExporterDisambiguation_GetFunding** job to call the Profiles Funding Disambiguation Engine web service to find grants for people.

Note: Make sure you load publications before searching for funding. The Funding Disambiguation Engine uses a person's list of publications to identify the correct funding items.

Loading Person Data: Part 6 – Convert data to RDF

Complete the data load process and convert data to RDF. Do this by opening ProfilesRNS_DataLoad_Part3.sql in SQL Management Studio, and then clicking the execute button to run the script. Note that this might take an hour or more to complete, depending on how much data is in your database.

Scheduling Database Jobs

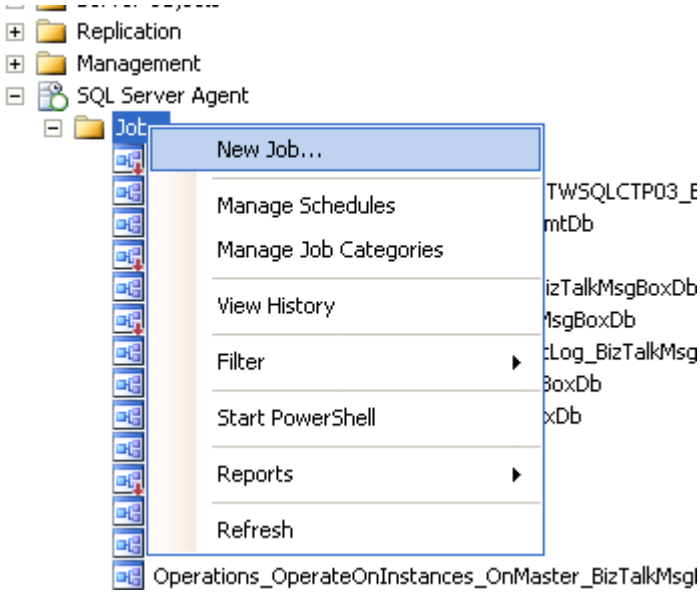
Follow the instructions in this section only if you are installing a new instance of Profiles RNS 2.12.0. Skip this section if you are upgrading from an existing Profiles RNS database.

Profiles RNS supports nightly updates of person data loaded into the import tables. In order to utilize this functionality, a set of database jobs must be scheduled to process the new data and generate the corresponding RDF. To create these jobs:

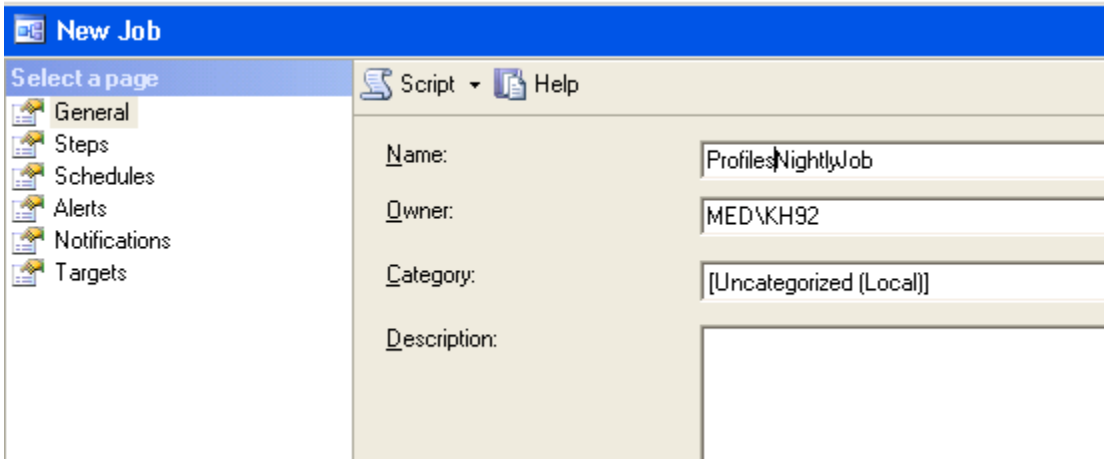
1) Schedule a job that runs “EXEC [Framework.].[RunJobGroup] @JobGroup = 4” nightly.
2) Schedule a job that runs “EXEC [Framework.].[RunJobGroup] @JobGroup = 5” weekly.
3) Schedule a job that runs “EXEC [Framework.].[RunJobGroup] @JobGroup = 6” monthly.

To create a sql agent Job, perform the following steps:

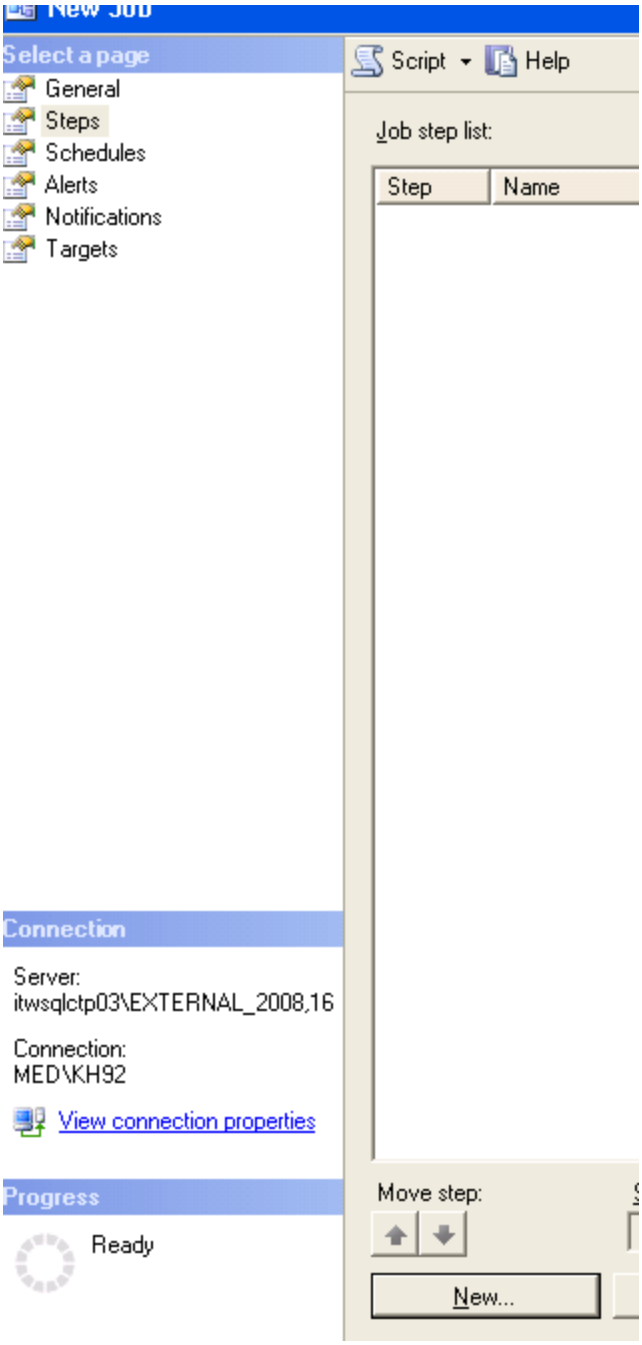
1. In SQL Manager, expand the SQL Server Agent Node and right click on the jobs folder, selecting “New Job”

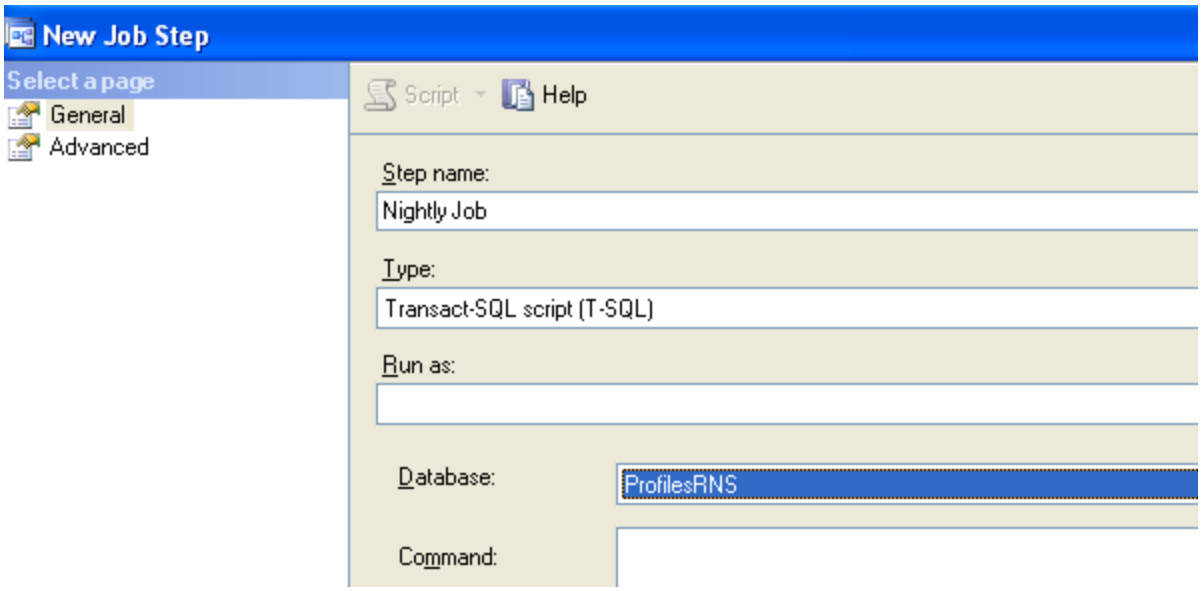


2. Create a name for the job

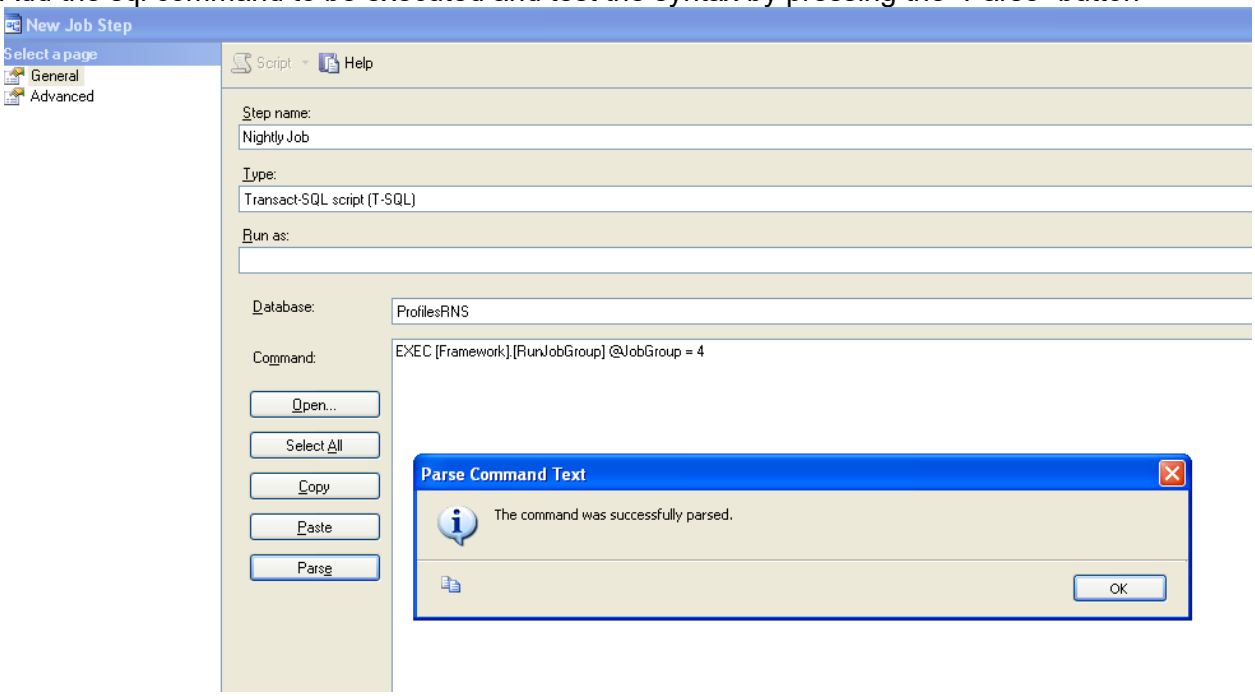


3. Select “steps” from the left pane and hit the “new” button.Create a step name and select “Transact-SQL script” as the job type





4. Add the sql command to be executed and test the syntax by pressing the “Parse” button



5. Hit OK to save changes through the remaining windows.

The [Framework].[RunJobGroup] procedure itself calls a number of other procedures that execute specific portions of the data load and update process. The [Framework].[Job] table lists these steps and indicates a status (“completed”, “processing”, or “error”) for each one.

NOTE: A step will only begin processing if all other steps have a “completed” status in the [Framework].[Job] table. If a step errors or hangs in a “processing” state, you must manually change the status value to “completed” before you can resume the scheduled jobs.

NOTE: The jobs described in this section replace the “Nightly”, “Weekly”, and “Monthly” jobs in older versions of Profiles RNS.

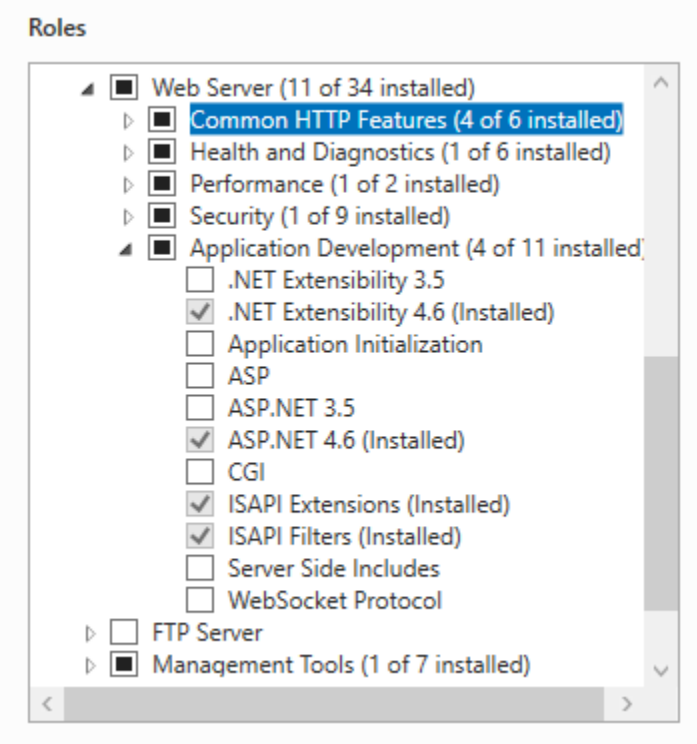
Configuring the Webserver

This section describes the minimum steps required to configure a webserver to run Profiles RNS. These instructions are based on installations using freshly installed server instances using Hyper-V and MSDN versions of Windows. Actual configuration steps may vary based on initial server configuration from manufacturer.

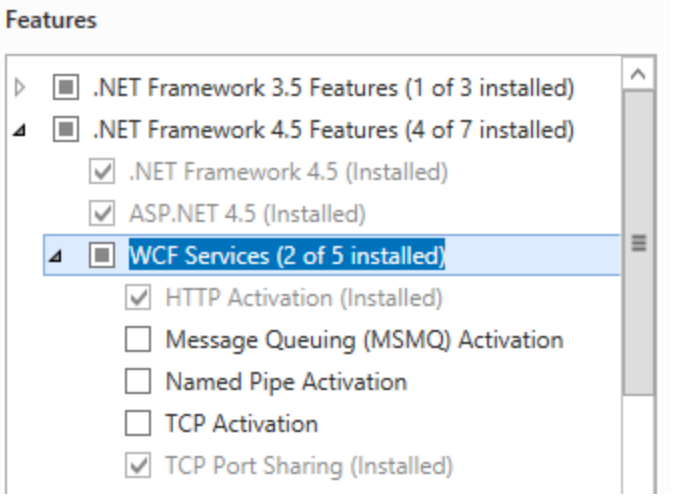
Windows Server 2016 and 2019

1. Install Web Server (IIS) Role using the “Add Roles and Features Wizard”.

2. On the roles page, ensure “.Net Extensibility 4.6” (“.Net Extensibility 4.7” for Server 2019), ISAPI Filters and ISAPI Extensions are selected. The .Net Framework will be installed automatically when this is selected.



3.On the features page expand “.NET Framework 4.6 / 4.7 Features”, then “WCF Services” then ensure “HTTP Activation” is selected.


Features
.NET Framework 3.5 Features (1 of 3 installed)
.NET Framework 4.5 Features (4 of 7 installed)
.NET Framework 4.5 (Installed)
ASP.NET 4.5 (Installed)
WCF Services (2 of 5 installed)
HTTP Activation (Installed)
Message Queuing (MSMQ) Activation
Named Pipe Activation
TCP Activation
TCP Port Sharing (Installed)

Installing the Website

Profiles has three components—a web application and two web services. They can be placed on the same IIS server instance or on different server instances, and they can be run as either websites or virtual directories.

This document does not cover custom security requirements and assumes the IIS defaults are used for public/Anonymous access.

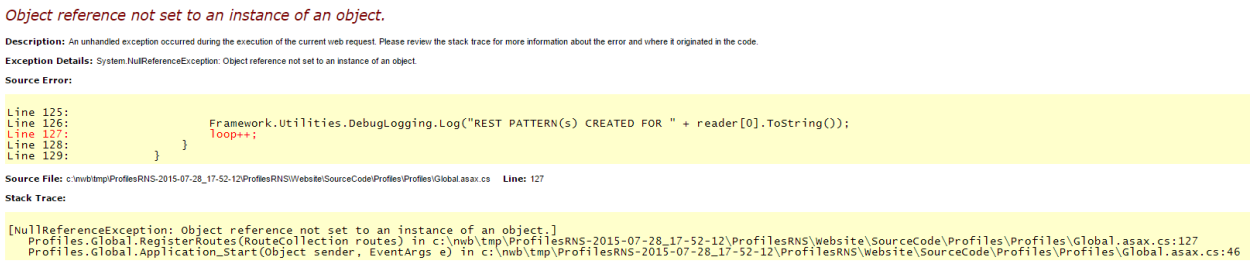
1) Open IIS Manager and create a virtual directory called Profiles and map its physical location to the drive and directory that will host the physical web files.
    - This step can be setup as a standalone website or sub web of existing website. Please consult your IT staff or IIS Administrator for what options are available to you if you are working on shared resources.
    - Please ensure that your web site or virtual directory is setup with execute scripts only access.
    - Ensure that your virtual directory is setup as an Application.
2) Copy the binary files from Website\Binary\Profiles to the physical location for hosting.
3) Update the ProfilesDB Connection string in the web.config file with your server, database, username and password.
4) Update the values for ProfilesRootRelativePath, and ProfilesRootURL keys. These must be set for your environment. If Profiles is hosted at a URL of https://example.com/profiles the values should be “profiles” and “https://example.com/profiles”. If instead profiles was hosted at https://example.com (at the root of the server), these values should be “” and https://example.com respectively.
5) From the web server file explorer, navigate to the physical location of the Profiles hosted files and provide the [ServerName]/IUSR account read access. Then provide the [ServerName]/IIS_IUSRS local server user account with read access to the same location.

Testing the Website

Before configuring any optional components it is important to test the website. The following two tests identify and provide troubleshooting suggestions for the three most common installation issues.

1. Go to the search page: Navigate to your profiles (e.g. http://example.com/profiles)

   - Success: Search page loads
   - Failure: Page redirects to http://localhost/profiles/search or another incorrect URL. BaseURI was not changed, or was set incorrectly while installing the database. See changing the BaseURI for details on fixing this error.
   - Failure: Object reference not set to an instance of an object.



Object reference not set to an instance of an object.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.NullReferenceException: Object reference not set to an instance of an object.

Source Error:

```
Line 125:
Line 126:                     Framework.Utilities.DebugLogging.Log("REST PATTERN(s) CREATED FOR " + reader[0].ToString());
Line 127:                     loop++;
Line 128:                 }
Line 129:             }
```

Source File: c:\nwb\tmp\ProfilesRNS-2015-07-28_17-52-12\ProfilesRNS\Website\SourceCode\Profiles\Profiles\Global.asax.cs **Line:** 127

Stack Trace:

```
[NullReferenceException: Object reference not set to an instance of an object.]
   Profiles.Global.RegisterRoutes(RouteCollection routes) in c:\nwb\tmp\ProfilesRNS-2015-07-28_17-52-12\ProfilesRNS\Website\SourceCode\Profiles\Profiles\Global.asax.cs:127
   Profiles.Global.Application_Start(Object sender, EventArgs e) in c:\nwb\tmp\ProfilesRNS-2015-07-28_17-52-12\ProfilesRNS\Website\SourceCode\Profiles\Profiles\Global.asax.cs:46
```

This error is caused by a failure to connect to the database. Confirm that the server, database, username and password are valid in the ProfilesDB connection string in your web.config file. Confirm that SQL server authentication is enabled on your SQL server instance. Confirm that remote connections are enables on SQL Server.

2. Search for a common keyword. Enter “Adult” in the keywords box and click search.

   - Success: This should take you to a search page with a number of results.
   - Failure: No results. Disambiguation has not been configured correctly. Review the “Loading Person Data: Part 4 – Obtaining Publications” section of this document.

Using the Website

Profiles RNS contains the following functionality:

- The New Search page has a form where users can type a keyword and search for matching RDF nodes. The default tab, “Find People”, only returns matching people. However, the search form on the “Find Everything” tab returns people as well as publications, concepts, organizations, or any other type of entity stored in the database. The Find Everything search results can also be narrowed to a single type using faceting. A “Mini-Search” box appears on the left sidebar on all pages other than the main Search form page.
- The About Profiles page is static text describing Profiles RNS.
- View RDF displays the RDF/XML for a profile, network, or connection page.
- Login allows users to login and edit their profiles or assign proxies.
- The SPARQL page, which is only available to members of the Admin security group, allows users to run an arbitrary search query against the database. It requires knowledge of how to construct SPARQL queries and interpret the results. An example query is placed in the search box by default.

Note that both the Search and SPARQL pages are tools to find URIs—the unique identifiers that characterize each RDF node in the database. The Search page is easy to use, but it restricts the user to certain types of queries. The SPARQL page allows any type of query, but it is only useful to certain types of users. The URIs in Profiles RNS have the form:

http://yourdomain.edu/profile/NodeID

Following URI/RDF conventions, this URI is simply an identifier. It does not return any content. If you enter the URI into a web browser, you will be redirected either to a URL that returns HTML content or RDF content, depending on the content-type in the request header. This process is called URI resolution. The corresponding HTML and RDF URLs are:

http://yourdomain.edu/display/NodeID

and

http://yourdomain.edu/profile/NodeID/NodeID.rdf

Once you are on the display page for a URI, the profile of the node will be rendered as HTML and will appear similar to how it looks in Profiles RNS Beta. To end-users the URI resolution will be seamless, and they will be able to navigate through pages in the same way as they do an ordinary website.

Additional Website Configuration and Optional Extensions

Logging and performance

The following settings can be configured in the Web.config file:

- appSettings/@key=DEBUG: If set to “true”, debugging information will be saved to a file named “ProfilesDebuggingLog.txt” at the root of the application directory.
- appSettings/@key=CACHE_EXPIRE: Sets the timeout in seconds for the cached IO requests.
- appSettings/@key=COMMANDTIMEOUT: Sets the timeout in seconds for database operations.
- appSettings/@key=aspnet:MaxHttpCollectionKeys: This is set to 10000 by default to enable the display of large tables in Profiles RNS.

Search Options

The search screen includes a number of dropdown filters. These are all displayed by default, and can be hidden if they are not relevant in your institution.

- appSettings/@key=ShowInstitutions: Set this to true if you want an Institution filter to appear on search forms.
- appSettings/@key=ShowDepartments: Set this to true if you want a Department filter to appear on search forms.
- appSettings/@key=ShowDivisions: Set this to true if you want a Division filter to appear on search forms.
- appSettings/@key=ShowOtherOptions: Set this to true if you want an Other Options filter to appear on search forms.

Google Analytics

Profiles will include Google Analytics javascript on every page if a tracking ID is included in the javascript.

- appSettings/@key= GoogleAnalytics.TrackingID: Set this to your Google Analytics tracking ID. Leave this key empty to remove the Google Analytics javascript.
- appSettings/@key=GoogleAnalytics.Domain: Sets the domain for Google Analytics tracking. Remove this key to for Google Analytics to use automatic domain tracking (recommended).
- appSettings/@key= GoogleAnalytics.TrackingID2: Optional, set this to add a second tracking ID.
- appSettings/@key= GoogleAnalytics.Domain2: Sets the domain for Google Analytics tracking for the second tracking ID. Remove this key to for Google Analytics to use automatic domain tracking (recommended).

Authentication

Profiles includes three authentication options. Profiles Authentication, Shibboleth and Active Directory.

Profiles Authentication

The inbuilt authentication in Profiles stores usernames and passwords in plain text in the database. There is no system for password maintenance. We recommend that production systems connect to their institutions authentication system rather than use the inbuilt authentication.

To use the inbuilt authentication modify the following web.config settings:

- `appSettings/@key=Login.PresentationXML`: Set this to “LoginFormPresentation” to use the inbuilt authentication

Load the username and password for each user into the username and password columns in the [User.Account].[User] table.

Shibboleth Authentication

The Shibboleth authentication module allows institutions with Shibboleth to use this to authenticate with Profiles. When authenticating with Shibboleth, a user who tries to login is redirected to the Shibboleth login page, where they enter their username and password. They are then redirected back to Profiles. As Profiles does not see the username entered by the user, shibboleth must be configured to return a user identifier to profiles in a header.

To use the Shibboleth authentication module, modify or add as needed the following web.config settings:

- `appSettings/@key=Login.PresentationXML`: Set this to “ShibbolethLoginPresentation” to use the shibboleth authentication
- `appSettings/@key=Shibboleth.ShibIdentityProvider`: Sets the identity provider
- `appSettings/@key=Shibboleth.UserNameHeader`: Set this value to the name of the shibboleth header that returns the user identifier to profiles
- `appSettings/@key= Shibboleth.LoginURL`: Set this value to the shibboleth login URL

Load the user identifier for each user into the username column in the [User.Account].[User] table. Leave the password column null for all users.

Active Directory Authentication

The Active Directory authentication module allows institutions to use active directory to authenticate with Profiles.

To use the Active Directory authentication module, modify or add as needed the following web.config settings:

- `appSettings/@key=Login.PresentationXML`: Set this to “ADLoginFormPresentation” to use the shibboleth authentication
- `appSettings/@key=AD.Domain`: Set this value to the domain to be queried
- `appSettings/@key=AD.User`: If your institution’s active directory security settings do not allow anonymous queries enter a username to use to query the directory. You will need to contact your AD administrator to get a username. Do not create this key if you are connecting anonymously.

- appSettings/@key=AD.Password: If your institution's active directory security settings do not allow anonymous queries enter a password to use to query the directory. You will need to contact your AD administrator to get a password. Do not create this key if you are connecting anonymously.
- appSettings/@key= AD.AccessContact: Set this value to an email address people can contact if they have problems logging in. This email will be shown as part of an error message to people who successfully authenticate with active directory, but so not have an account in profiles.

Load the active directory username for each user into the username column in the [User.Account].[User] table. Leave the password column null for all users.

Maps

Google Maps with display with a "For development purposes only" watermark until an API key is added. Create an API key as described in the Geocoding section of this document, and enter this key into the appSettings/@key=GoogleMapsKey: setting in your web.config.

When displaying Google Maps, Profiles lets the user select from several preset zooming and centering configurations. For example, by default, Harvard's Profiles shows the city of Boston, but users can click a link to zoom out to show all of New England. Enter your list of map presets in the Profiles/Profile/Modules/GoogleMap/config.xml file. (This is a module-specific configuration file.) Each preset is defined in an xml <Zoom> node and one of these presets should be flagged as DefaultLevel = "True".

DIRECT2Experts

DIRECT2Experts (http://direct2experts.org) is a federated query tool that searches more than 40 institutions using 8 different research networking products. On the search results page in the Profiles RNS website, when users click "search other institutions", the website is using DIRECT2Experts. If you want your instance of Profiles RNS to be discovered by other institutions using DIRECT2Experts, then make the following changes to config.xml and DIRECT.xml:

a. Edit the config.xml file located in Profiles/DIRECT/Modules/SearchInstutions/. This file is used to display the institution description when a user hovers their mouse over the results table of a federated query and sets the timeout value for an external query to an institution.

```
<directconfig>
  <directpopulationtype>[INSTITUTION DESCRIPTION HERE]</directpopulationtype>
  <querytimeout>8</querytimeout>
</directconfig>
```

b. Edit the DIRECT.xml file located in the root of Profiles/Direct/. This file is the bootstrap file used to publish your Institution name and Query Endpoint for the federated search.

```
<site-description>
  <name>[INSTUTION NAME HERE]/name>
  <aggregate-query>http://[DOMAIN NAME HERE]/[SUB WEB NAME HERE]/
    DIRECT/Modules/SearchInstitutions/
    DirectService.asp?Request=IncomingCount&amp;SearchPhrase=</aggregate-query>
</site-description>
```

c. Email the full URL of your DIRECT.xml file to profiles@hms.harvard.edu, and we will add your site to DIRECT2Experts.org.

d. In order to modify the list of institutions that are searched when users click the "search other institutions" link in your Profiles RNS website, edit the data in the [Direct.].[Sites] table.

Group Profiles

Group Profiles is feature that creates separate pages in the Profiles RNS website for centers, laboratories, projects, or other groups of people. This allows a group to share information about itself on the Profiles RNS website and link to the profile pages of its members.

Group Profiles – Background

Group Profiles was developed as a collaboration between the Harvard Profiles RNS team and Christopher Shanahan and Chris Dorney from the Boston University Clinical Translational Science Institute (CTSI). In Phase 1 of the Group Profiles project, we implemented the minimal basic functionality needed to demonstrate each of the key features. We plan to expand the functionality of Group Profiles in the future and hope that other institutions also contribute new modules.

Group Profiles – Key Features

- Membership. Managers of Group Profile pages can affiliate individuals within the Profiles RNS website as members of the group. The list of members are shown on the Group Profile page; and, the groups a person is affiliated with are shown on his or her profile page in a My Groups module.
- Custom Page Layout. Within the Profiles RNS framework, Group Profiles are a distinct object class, which means they can have different content modules and page layout than person profile pages.
- Custom Modules. Certain Group Profile modules enable group managers to post content directly to the Group Profile pages. Other modules can be configured to pull content (automatically or manually) from member profile pages and display it on the Group Profile page. The current list of modules include: (a) Photo (group logo, picture, etc.); (b) Welcome, About Us, and Contact text; (c) Open Social Gadgets (e.g., links to other groups or external websites); and (d) Publications.
- Security. Group Profiles utilize the Profiles RNS security model, which enables group managers to control whether content is visible to the public or restricted to certain users. All Groups have an End Date, after which the group will automatically be hidden from the website. Group Managers can set or change the End Date as needed.
- Search. Group Profile page content is searchable through the website and the Profiles RNS APIs.
- Open Data. Group Profile page content is integrated into the Profiles RNS semantic web architecture so that it can be exchanged with other applications (like VIVO) via linked open data.

Group Profiles – Roles

There are three types of user roles related to Group Profiles:

- Member. A person associated with a group. A member must be a user, but doesn't need to have a profile. Members can be given custom titles, such as "Director" or "Contact Person". VIVO RDF is used to link the user to the group. (In the future, users will be able

to request to join a group through a web interface, but it must be approved by a manager or admin. In Phase 1, users can email a group manager.)

- Manager. A person who can add/remove group members, change group visibility permissions, and post content. A manager cannot change the name of a group. Managers are assigned to specific groups by admins or other managers through a web-based interface. RDF is not generated for a manager. In Phase 1, managers must email admins to request new groups. Managers will see an "Edit this Profile" link on their left sidebar when they are viewing their group's Profiles page. They can click this link to add/remove members and post content.
- Admin. A person who can create, delete, and rename groups. Admins have manager rights to all groups. Admins are created by manually entering users IDs into a database table. RDF is not generated for an admin. The Admin interface (creating, deleting, renaming groups) is a separate application, accessed from a "Manage Groups" link on the left sidebar, under "Manage Proxies".

The basic workflow for creating a new Group Profile is (1) a person emails an admin requesting a new group, (2) the admin creates the group and assigns a user manager rights to that group, (3) the manager can give additional users manager rights to the group, (4) the manager adds members to the group and posts content.

Group Profiles – Initial Setup

Group Profiles is automatically enabled. In order to create groups, the UserIDs of one or more group administrators must be manually entered into the `[Profile.Data].[Group.Admin]` table. When group administrators login to Profiles, a Manage Groups link will be available on the left sidebar, which they can use to setup new groups and assign managers.

Group Profiles - Ontology

Group Profiles use the FOAF Group class, with a combination of VIVO and PRNS properties. The VIVO MemberRole class is used to link groups to people. A new PRNS property, associatedInformationResource, links groups directly to publications (without going through an Authorship entity). Below is an outline of the key classes and properties:

- http://xmlns.com/foaf/0.1/Group
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#type
    - http://xmlns.com/foaf/0.1/Group
    - http://xmlns.com/foaf/0.1/Agent
  - http://www.w3.org/2000/01/rdf-schema#label
    - The name of the group
  - http://profiles.catalyst.harvard.edu/ontology/prns#hasGroupOptions
    - Placeholder property to enable editing of group's view permissions & end date
  - http://profiles.catalyst.harvard.edu/ontology/prns#hasGroupManager
    - Direct link to manager's URI
  - http://vivoweb.org/ontology/core#contributingRole
    - Link to member through a MemberRole object
  - http://profiles.catalyst.harvard.edu/ontology/prns#associatedInformationResource
    - Direct link to a publication
  - http://profiles.catalyst.harvard.edu/ontology/prns#mainImage
  - http://vivoweb.org/ontology/core#overview
- http://vivoweb.org/ontology/core#MemberRole

        - http://www.w3.org/1999/02/22-rdf-syntax-ns#type
            - http://vivoweb.org/ontology/core#MemberRole
            - http://vivoweb.org/ontology/core#Role
        - http://www.w3.org/2000/01/rdf-schema#label
            - “Member” (default), “Director”, “Alumni”, etc.
        - http://vivoweb.org/ontology/core#memberRoleOf
            - Link to Person
        - http://vivoweb.org/ontology/core#roleContributesTo
            - Link to Group
- http://xmlns.com/foaf/0.1/Person
    - http://vivoweb.org/ontology/core#hasMemberRole
        - Link to MemberRole, which links to a Group
- http://vivoweb.org/ontology/core#InformationResource
    - As usual, but without a link to an author or the group

ORCID Extension

Profiles contains two ORCID modules.

- Basic ORCID Module
- ORCID integration Module

The ORCID integration module is intended to provide a fully featured ORCID integration for institutions who are ORCID members. The basic ORCID module allows users to add an ORCID to their profile, but does not include any further features. This module can be used by institutions who are not ORCID members.

For further details about ORCID see http://orcid.org/about/what-is-orcid.

Basic ORCID Module

The Basic ORCID Module can be enabled by running the following SQL query:

```
update [Ontology.].ClassProperty
       set IsDetail = 0,
       CustomDisplay = 1,
       CustomEdit = 1,
       ViewSecurityGroup = -1,
       EditSecurityGroup = -20,
       EditPermissionsSecurityGroup = -20,
       EditExistingSecurityGroup = -20,
       EditAddNewSecurityGroup = -20,
       MaxCardinality = 1,
       CustomEditModule = '<Module ID="CustomEditBasicORCID" />'
       where class = 'http://xmlns.com/foaf/0.1/Person'
       and property = 'http://vivoweb.org/ontology/core#orcidId'
```

ORCID Integration Module

The ORCID Integration Module users an API that is no longer supported by ORCID. We are working on restoring the functionality of this module to Profiles in future version.

Customizing Profiles

The profiles application can be modified using Microsoft Visual Studio. Open the Website\SourceCode\Profiles\Profiles.sln solution in Visual Studio to edit the source code. The Profiles architecture is described in the ProfilesRNS Architecture Guide, and an example of adding custom data into the Profiles ontology is included in the SQL_Examples\ProcessDataMap.sql file.

Installing the APIs

The Profiles APIs can be placed on the same IIS server instance or on different server instances from the website. They can be run as either websites or applications within a single website.

If you are hosting the Profiles web application and Profiles web services on different servers, you will need to repeat the instructions in “Configuring the Web Server” on the additional server.

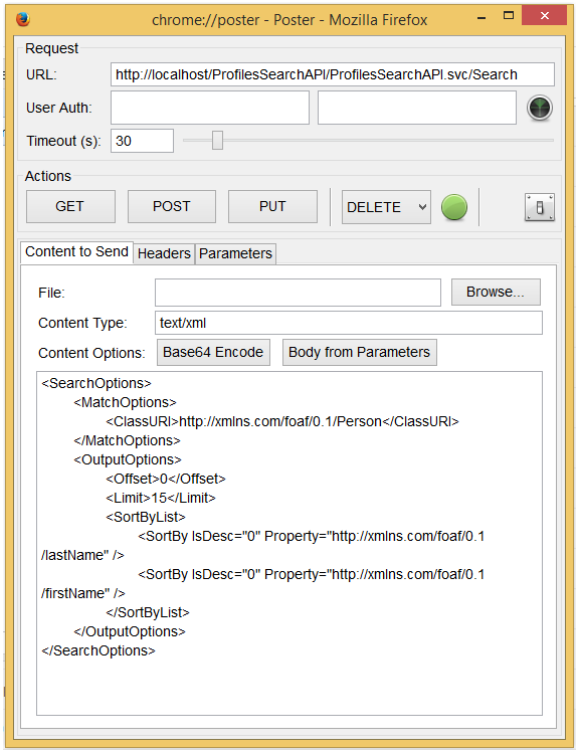
1) Open IIS Manager and create a virtual directory called ProfilesSearchAPI and map its physical location to the drive and directory that will host the physical web files.
    - This step can be setup as a standalone website or sub web of existing website. Please consult your IT staff or IIS Administrator for what options are available to you if you are working on shared resources.
    - Please ensure that your web site or virtual directory is setup with execute scripts only access.
    - Ensure that your virtual directory is setup as an Application.
2) Copy the binary files from Website\Binary\ProfilesSearchAPI into the physical location for hosting.
3) From the web server file explorer, navigate to the physical location of the ProfilesSearchAPI hosted files and provide the [ServerName]/IUSR account read access. Then provide the [ServerName]/IIS_IUSRS local server user account with read access to the same location.
4) Update the ProfilesDB Connection string in the web.config file with your server, database, username and password.
5) Repeat Steps 1 – 3 for ProfilesSPARQLAPI
6) Update the SemWebDB connection string in the web.config. Provide the server name, database name, userid and password for the default connection string. Note that in the sqlserver parameter, the database name must be enclosed by brackets.
7) Update the Profiles web.config. Set `appSettings/@key=SPARQLEndPoint` to the full absolute URL of the SPARQL API (http://[yourdomain]/ProfilesSPARQLAPI.svc/Search).
8) Optional: The Profiles RNS Beta website had a web service API, which has changed in Profiles RNS 1.0. If you need to continue to support that API, then replace its code with the Connects.Profiles.Service project files located in the ProfilesBetaAPI directory. In the web.config file, use the same isSecure value (true or false) that you were previously using, but change the connection string to point to the Profiles RNS database.

Testing the APIs

To test the APIs, you will need to post xml queries to them. There are many applications that enable this, the screenshots in this section are using the firefox addon “Poster”. In each of these tests you will send a query for all people in the database, and confirm that the correct number of people are returned.

To test the Profiles Search API post the following XML to your search API URL. (e.g. http://[yourdomain]/ProfilesSearchAPI/ProfilesSearchAPI.svc/Search)

```
<SearchOptions>
    <MatchOptions>
        <ClassURI>http://xmlns.com/foaf/0.1/Person</ClassURI>
    </MatchOptions>
    <OutputOptions>
        <Offset>0</Offset>
        <Limit>15</Limit>
        <SortByList>
            <SortBy IsDesc="0" Property="http://xmlns.com/foaf/0.1/lastName" />
            <SortBy IsDesc="0" Property="http://xmlns.com/foaf/0.1/firstName" />
        </SortByList>
    </OutputOptions>
</SearchOptions>
```



The response should be an XML document, The value of the prns:numberOfConnections should match the number of people loaded into your database.

To test the Profiles SPARQL API post the following XML to your SPARQL API URL. (e.g. http://[yourdomain]/ProfilesSPARQLAPI/ProfilesSPARQLAPI.svc/Search)

<query-request><query>PREFIX core: <http://vivoweb.org/ontology/core#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT DISTINCT ?s

WHERE {

  ?s rdf:type foaf:Person

}</query></query-request>

The result will be an XML document containing the URL of each person in the database. Check that the number of results matched the number of people you loaded into the database.

Using the APIs

Use of the APIs is described in the ProfilesRNS_APIGuide included with Profiles.

Additional API Configuration

The ProfilesSearchAPI has the following configuration options in the web.config file:

- `appSettings/@key=DEBUG`: If set to “true”, debugging information will be saved to a file named “ProfilesDebuggingLog.txt” at the root of the application directory.
- `appSettings/@key=CACHE_EXPIRE`: Sets the timeout in seconds for the cached IO requests.
- `appSettings/@key=COMMANDTIMEOUT`: Sets the timeout in seconds for database operations.
- `appSettings/@key=SecurityMode`: Set to “Public” if this API should use the “Public” search cache and only return data intended for the general public. Set to “Private” if this API should use the “Private” search cache and return all data in Profiles RNS that can be accessed by the “harvester” security group.

The ProfilesSPARQLAPI has the following configuration options in the web.config file:

- `connectionStrings/@name=SemWebDB`: Provide the server name, database name, userid and password for the default connection string. Note that in the sqlserver parameter, the database name must be enclosed by brackets.
- `appSettings/@key=LogService`: If set to “true”, information about all data IO requests. Ensure this is set to false after testing as production volume will fill this log up fast. The root folder must have write permissions for the log file to be created.
- `appSettings/@key=CACHE_EXPIRE`: Sets the timeout in seconds for the cached IO of SPARQL requests.
- `appSettings/@key=COMMANDTIMEOUT`: Sets the timeout in seconds for database operations.
- `appSettings/@key=SecurityMode`: Set to “Public” if this API should use the “Public” SemWeb views and only return data intended for the general public. Set to “Private” if this API should use the “Private” SemWeb views and return all data in Profiles RNS.

Optional: If you want to have different versions of the ProfilesSearchAPI or ProfilesSPARQLAPI with different SecurityMode settings, then you will need to duplicate the entire API as a new .NET application and modify the web.config file. A common approach is to have a public API than anyone can use, and a private API that uses IP address restrictions in IIS to limit the API to known applications.