Profiles Research Networking Software Installation Guide

Documentation Version: July 25, 2014

Software Version: ProfilesRNS_2.1.0

Table of Contents

Introduction

This document describes how to install the Profiles RNS 2.1.0 database, website, and APIs. This version of Profiles RNS is an intermediate release to make the latest version of the OpenSocial/ORNG code available to the public as quickly as possible. It will be followed by version 2.5.0, which includes integration with ORCID, in early August 2014. This, in turn, will be followed by version 2.5.1 in late August, which will have performance enhancements and upgrade paths from previous versions of Profiles RNS. **We recommend continuing to use version 2.0.0 or waiting for version 2.5.1, rather than installing this version of Profiles RNS.**

Hardware and Operating System Requirements

Profiles RNS is a Microsoft .NET 3.5 website that uses a Microsoft SQL Server 2012 (or 2005, 2008, 2008 R2) 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 then 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.

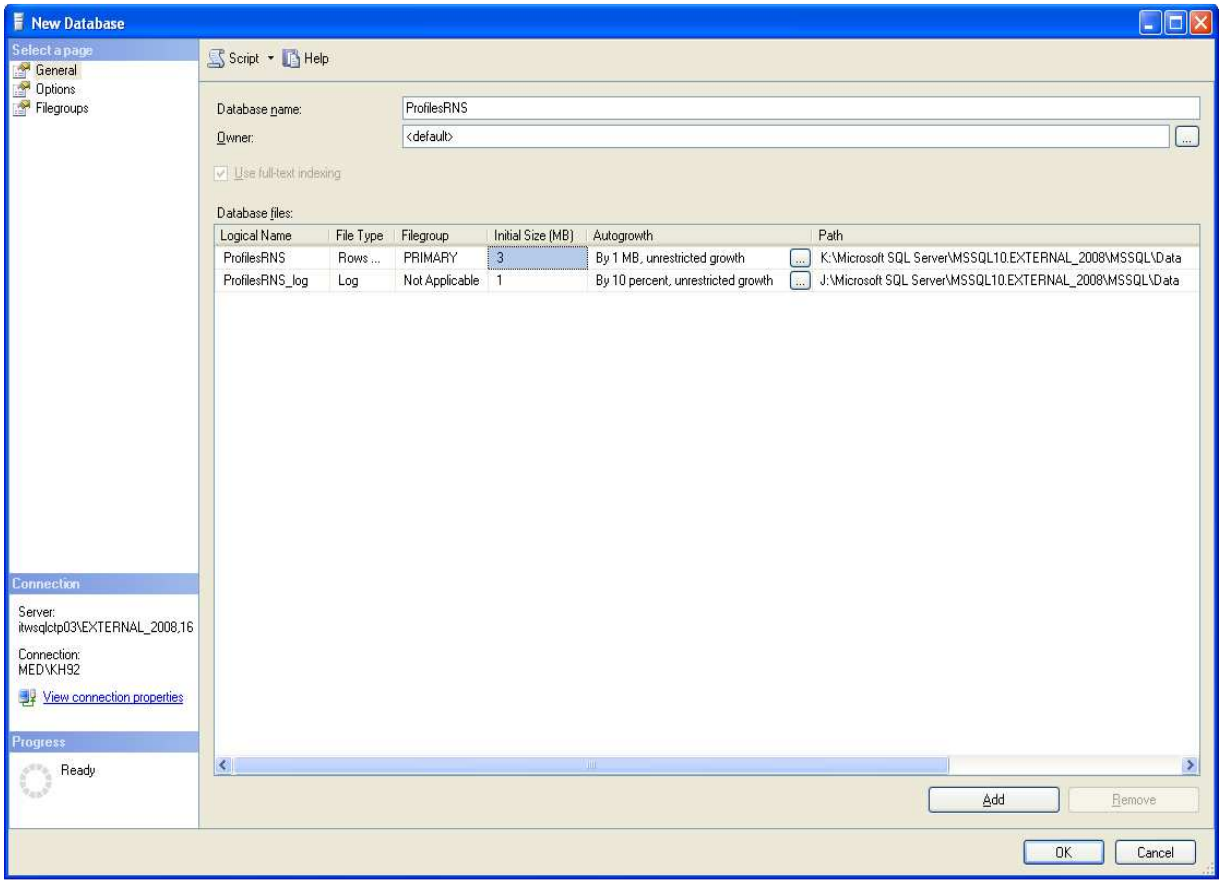
2) GitHub Clone – The latest source code can be cloned from https://github.com/ProfilesRNS/ProfilesRNS. A GitHub clone gives developers access to the latest Profiles RNS development branch. This code may be unstable and should not be used in production systems.

Installing the Database

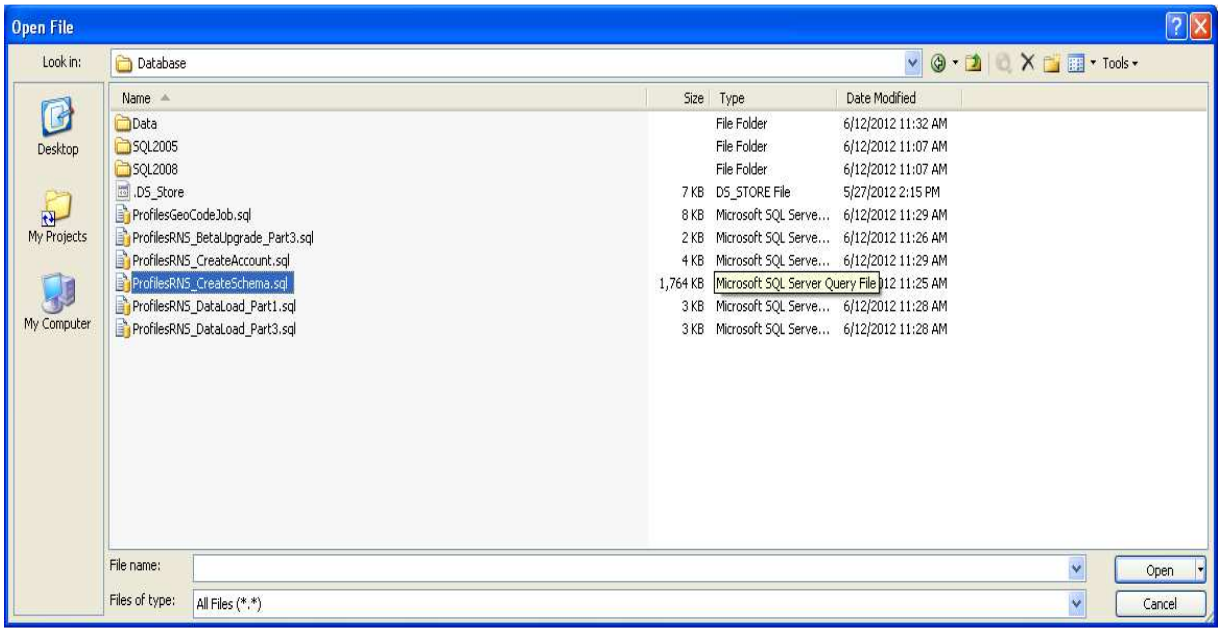
Follow the instructions in this section only if you are installing a new instance of Profiles RNS 2.1.0 Skip this section if you are upgrading from an existing Profiles RNS 2.0.0 database.

Follow the steps below to create a new database:

1. Create a new empty database.
    a) In SQL Management Studio, right-click on the databases folder underneath your SQL Server Instance name and select “New Database”.
    b) Once you see the screen below, choose a name for your new profiles database (e.g., ProfilesRNS) and storage location for its data files.

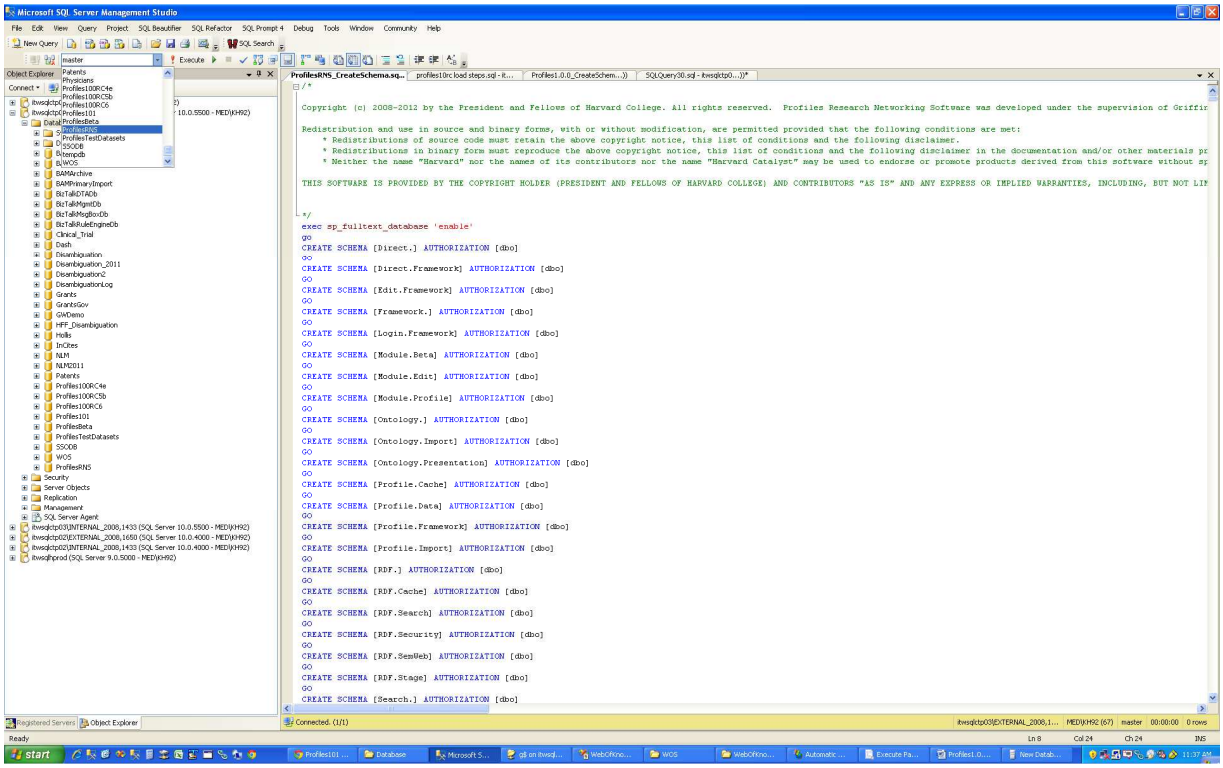


2. Add objects tables, stored procedures, and other objects to the new database.
    a) In SQL Management Studio, open the ProfilesRNS_CreateSchema.sql file.



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) Select the ProfilesRNS database.
    c) 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 six data files (InstallData.xml, VIVO_1.4.owl, PRNS_1.0.owl, SemGroups.xml, MeSH.xml, and ORNG_1.0.owl). 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

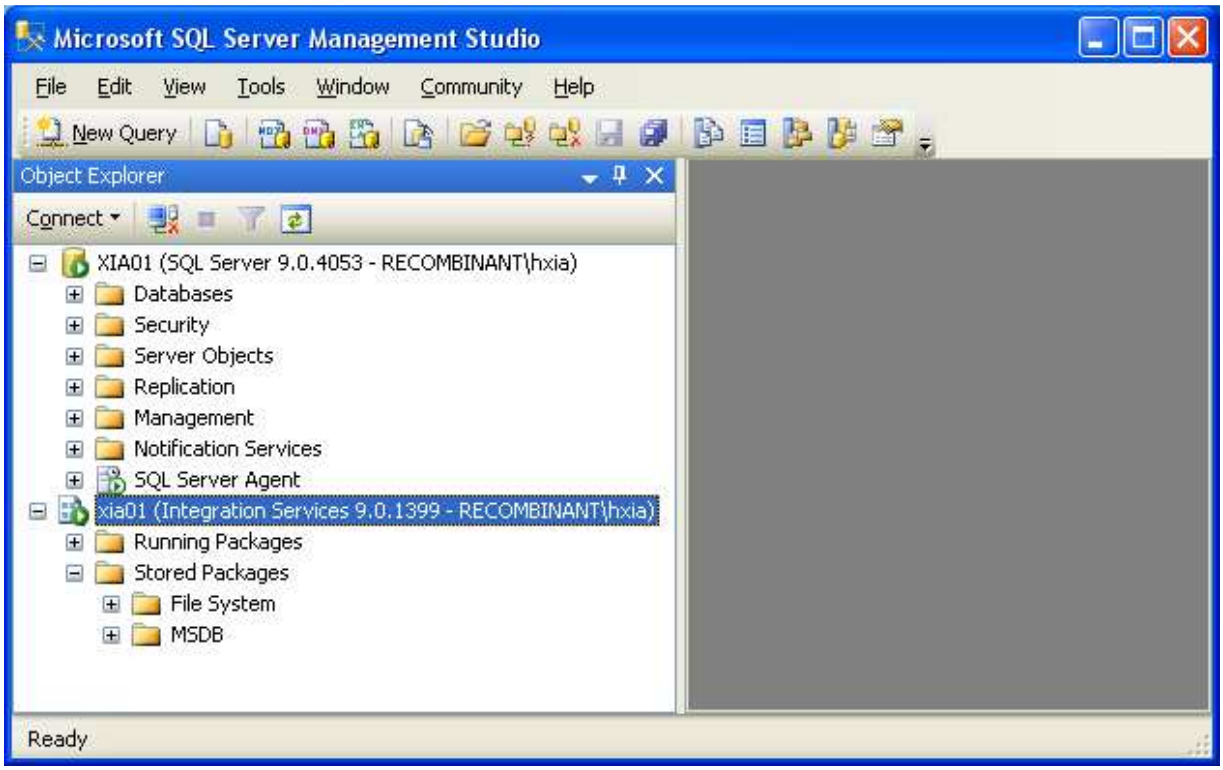
Follow the instructions in this section only if you are installing a new instance of Profiles RNS 2.1.0. Skip this section if you are upgrading from an existing Profiles RNS database.

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

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

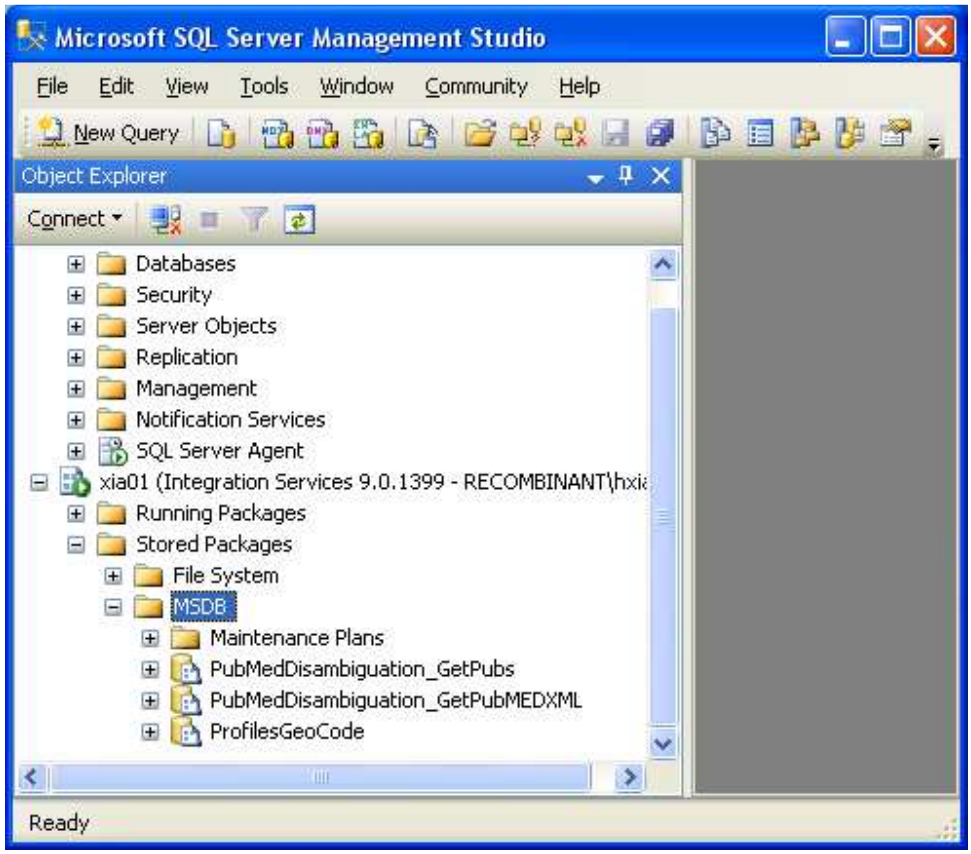
   iii. Left click **Package name** field and the package name will be filled automatically

   iv. Click **Ok** to install

   You can use this procedure to install the following three packages. Note that there are different versions of these packages for SQL Server 2005 and SQL Server 2008. The 2005 packages are in a folder named SQL2005, and the 2008 packages are in a folder named SQL2008.

   1. PubMedDisambiguation_GetPubs.dtsx
   2. PubMedDisambiguation_GetPubMEDXML.dtsx
   3. ProfilesGeoCode.dtsx

   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. ProfilesGeoCodeJob.sql
    b. PubMedDisambiguation_GetPubs.sql
    c. PubMedDisambiguation_GetPubMEDXML.sql

   For each of the scripts, modify the following parameters in the sql code:
    - @owner_login_name – the name of the sysadmin account created in step #4.
    - @database_name – the name of the profiles database.
    - @server_name – the name of the sql server instance.

   Note that there are three versions of PubMedDisambiguation_GetPubs.sql and PubMedDisambiguation_GetPubMEDXML.sql, depending on whether you are using SQL Server 2005, 2008 or 2012. The 2005 files are in the folder SQL2005, the 2008 files are in the folder SQL2008, and the 2012 files are in the SQL2012 folder.

3. The following scripts that call SSIS Packages need to be modified so that they work in your particular environment:
    a. ProfilesGeoCodeJob.sql
    b. PubMedDisambiguation_GetPubs.sql
    c. PubMedDisambiguation_GetPubMEDXML.sql

   For each of the scripts listed in step #6, 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.

4. 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

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

In order for Profiles to display the locations of faculty on a map:

1. Profiles must convert person addresses to latitude and longitude coordinates. It uses the addressstring column in the person table for this purpose. Thus, this field must be populated in the [Profile.Import].[Person] loading table to display people on Google Maps. The value for addressstring should be a street address, with no additional information. For example, a valid addressstring is “25 Shattuck Street, Boston, MA 02115”. An invalid addressstring is “Harvard Medical School, Information Technology, 25 Shattuck Street, Room 101a, Box 12, Boston, MA 02115”.
2. After you have completed loading the person data, run the **ProfilesRNSGeoCode** job. 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 a 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

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%
```

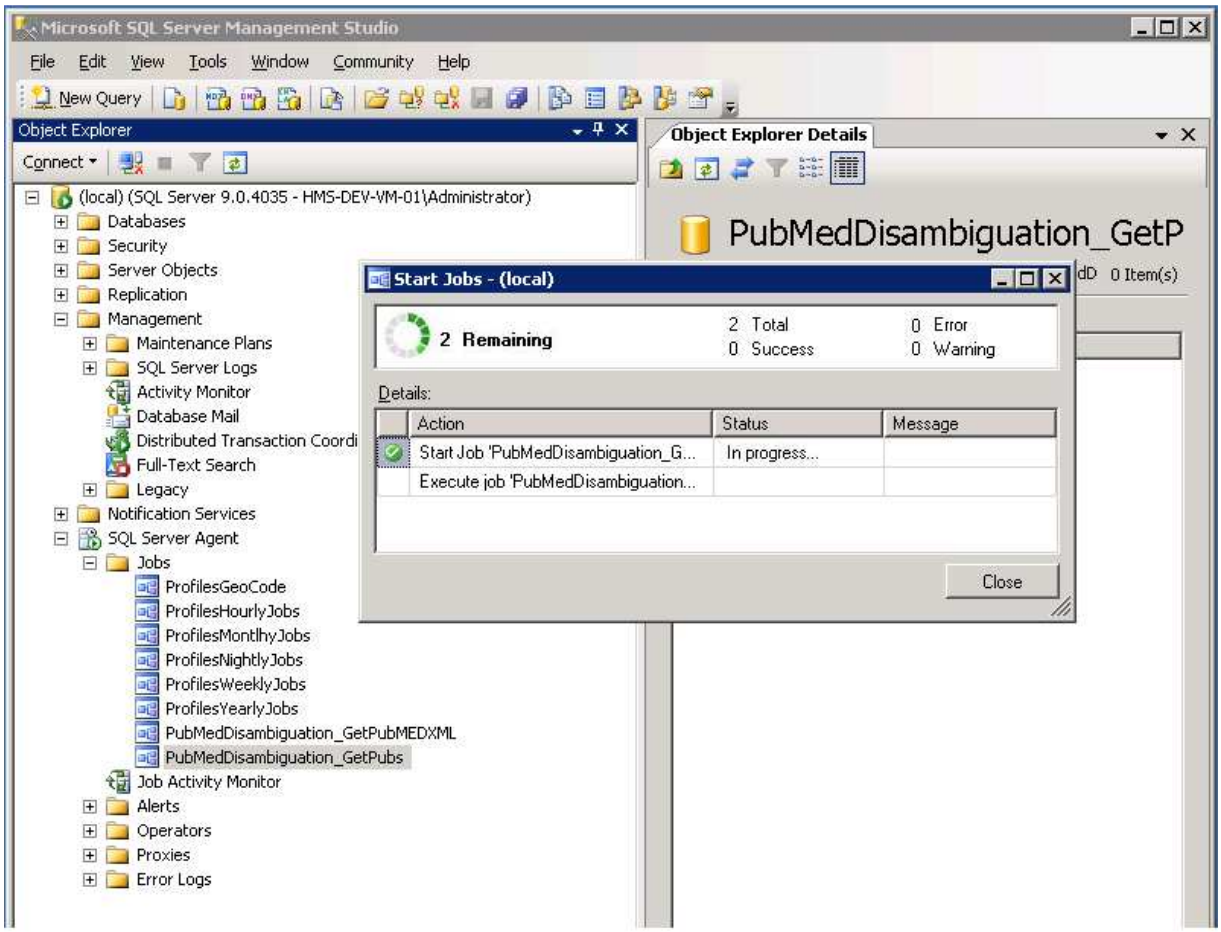
Example:

```
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%')
```

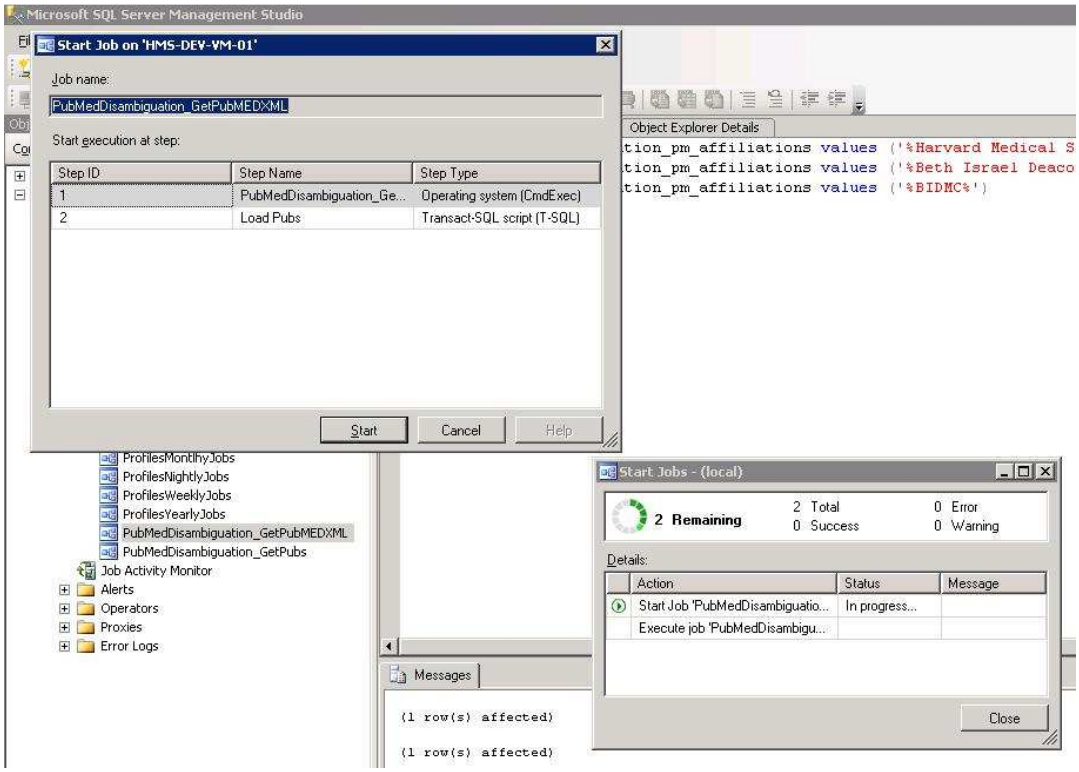
Examples of strings that we do <u>not</u> use at Harvard because they would be too broad are:

```
%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.



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 – 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

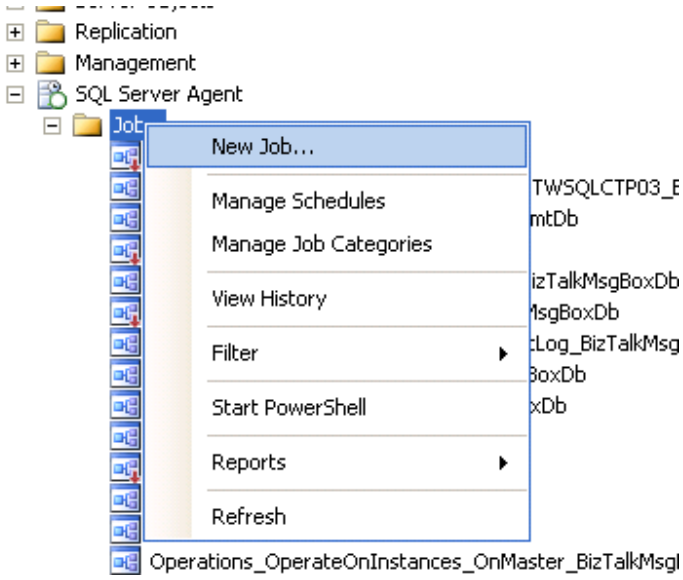
Follow the instructions in this section only if you are installing a new instance of Profiles RNS 2.1.0. Skip this section if you are upgrading from an existing Profiles RNS database.

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:

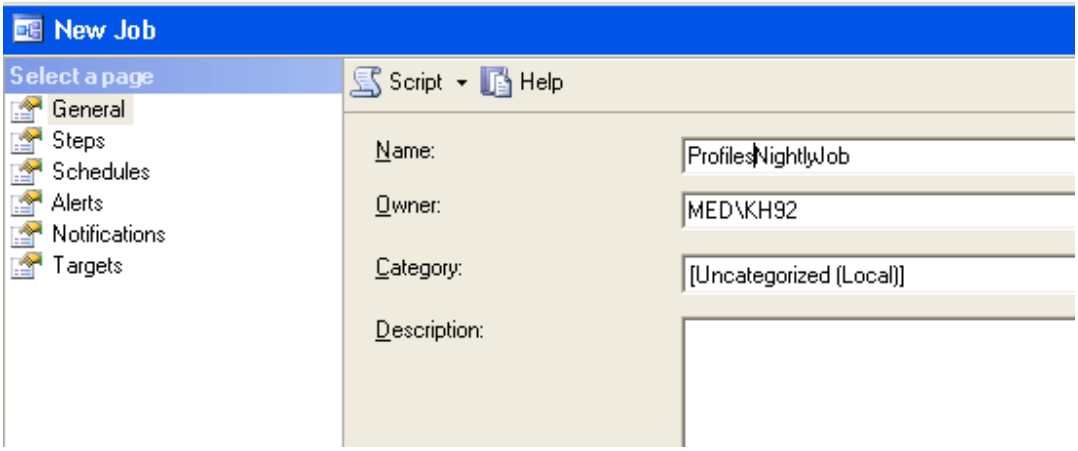
1) Schedule a job that runs “EXEC [Framework.].[RunJobGroup] @JobGroup = 4” nightly.
2) Schedule a job that runs “EXEC [Framework.].[RunJobGroup] @JobGroup = 5” weekly.
3) Schedule a job that runs “EXEC [Framework.].[RunJobGroup] @JobGroup = 6” monthly.

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

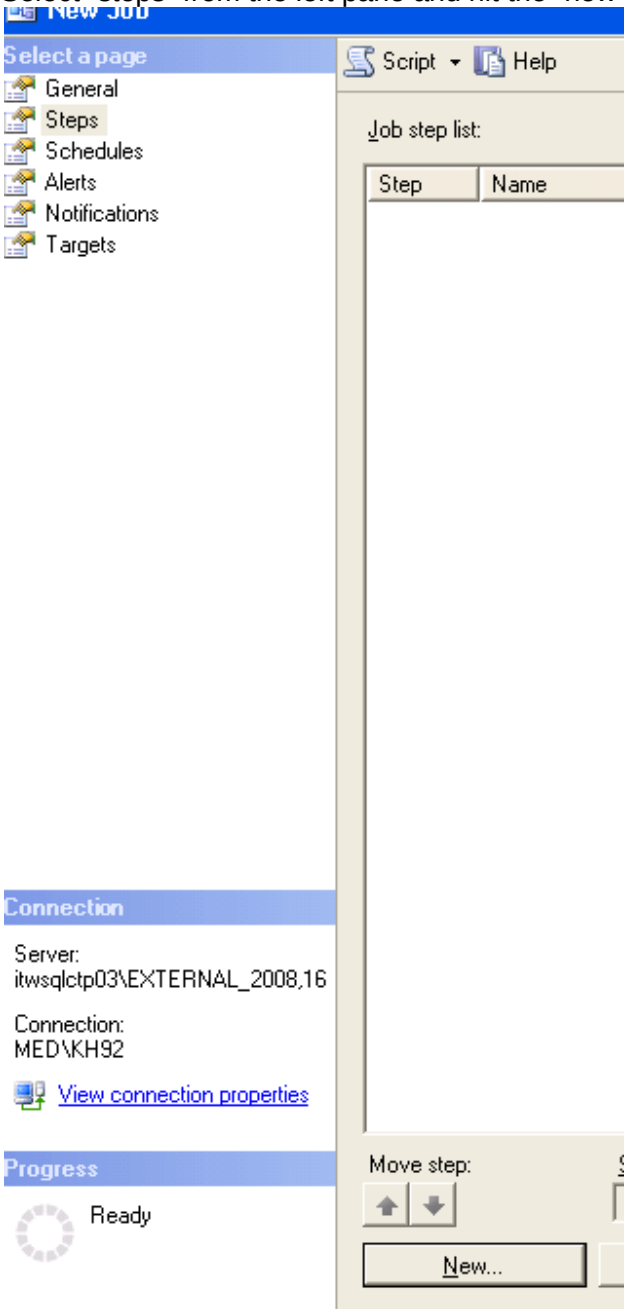
1. In SQL Manager, expand the SQL Server Agent Node and right click on the jobs folder, selecting “New Job”



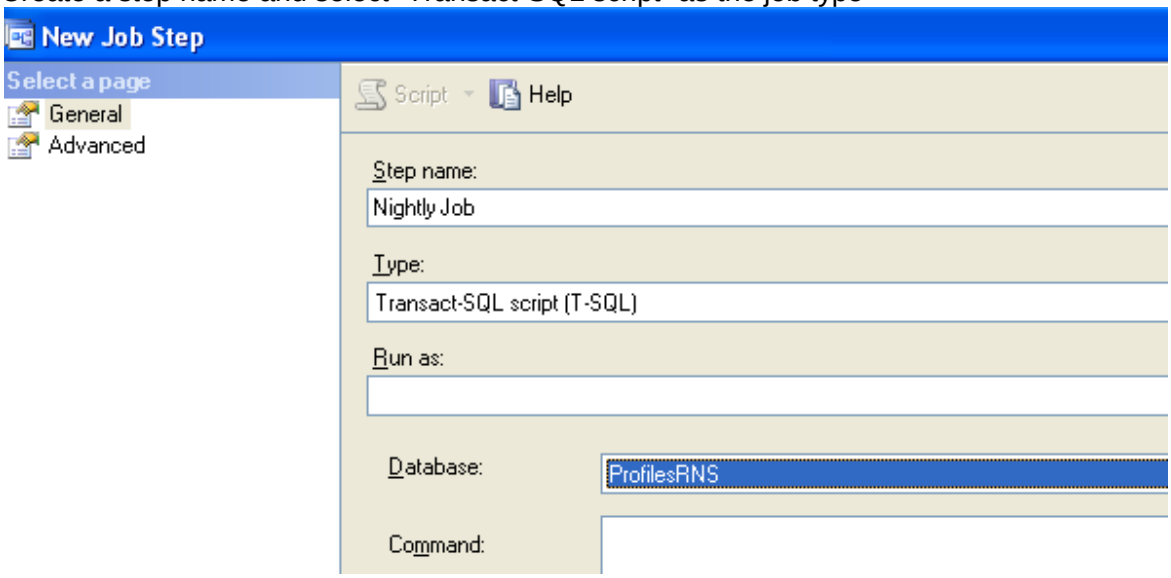
2. Create a name for the job


New Job
Select a page
General
Steps
Schedules
Alerts
Notifications
Targets
Script
Help
Name:
ProfilesNightlyJob
Owner:
MED\KH92
Category:
[Uncategorized (Local)]
Description:

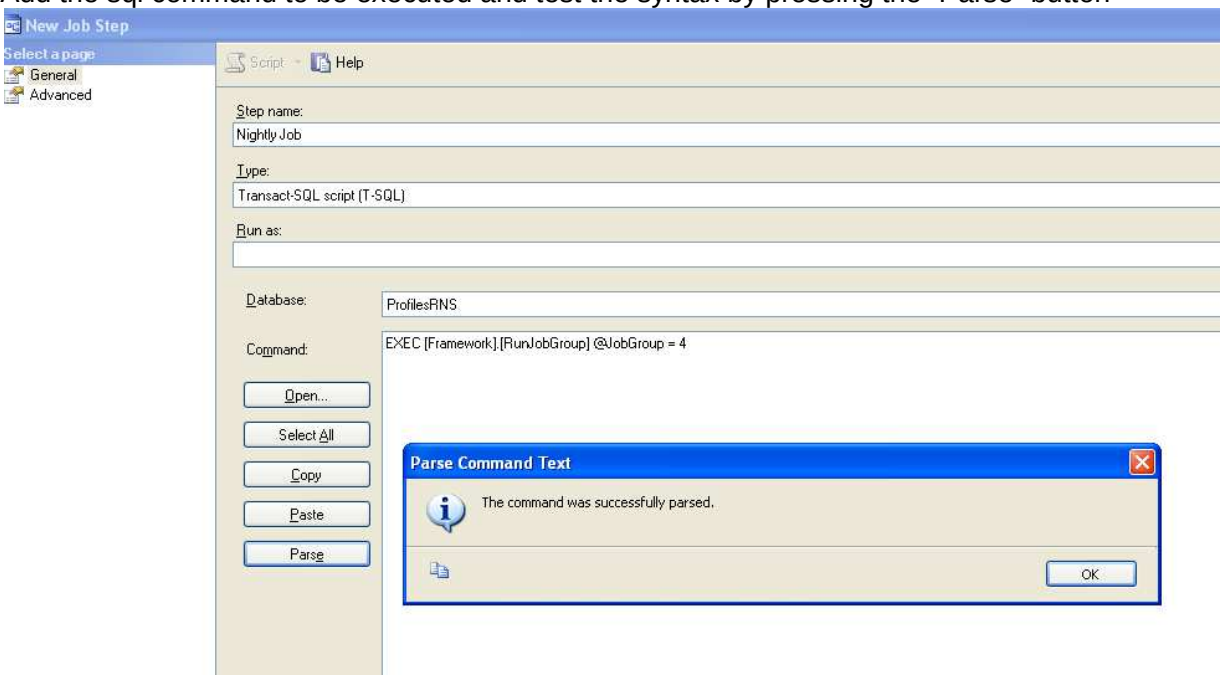
3. Select “steps” from the left pane and hit the “new” button.



4. Create a step name and select “Transact-SQL script” as the job type



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



6. 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.

Installing the Code

Follow the instructions in this section if you are installing a new instance of Profiles RNS 2.1.0 or upgrading from an older version of Profiles RNS.

The four main components are the Profiles web site, the Profiles SPARQL API, the Profiles Search API and the source code for the SemWeb library.

Solution name:
Profiles

Web application:
Profiles

Web service:
ProfilesSPARQLAPI
ProfilesSearchAPI

Installation steps:

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. If you are hosting the Profiles web application and Profiles web services on different servers, then certain steps listed below will need to be performed on both servers. This document does not cover custom security requirements and assumes the IIS defaults are used for public/Anonymous access.

1) Ensure Microsoft .Net version 3.5 is installed on all web servers used to host the profiles system.
2) If this is the first ASP.NET application you are running on your server, you need to register ASP.NET in IIS (Depending on a 32 bit or 64 bit app pool):
    - On a 32-bit OS, from the web server command prompt run the following utility, including the full path:
      `C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis.exe –i`
    - On a 64-bit OS, from the web server command prompt run the following utility, including the full path:
      `C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_regiis.exe –i`
3) Open the web server file explorer and provide the [ServerName]/IIS_WPG local server user account with read/write permissions for the "C:\Windows\Microsoft.NET\Framework" directory and subdirectories.
4) Open IIS version 6.0 or greater 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.

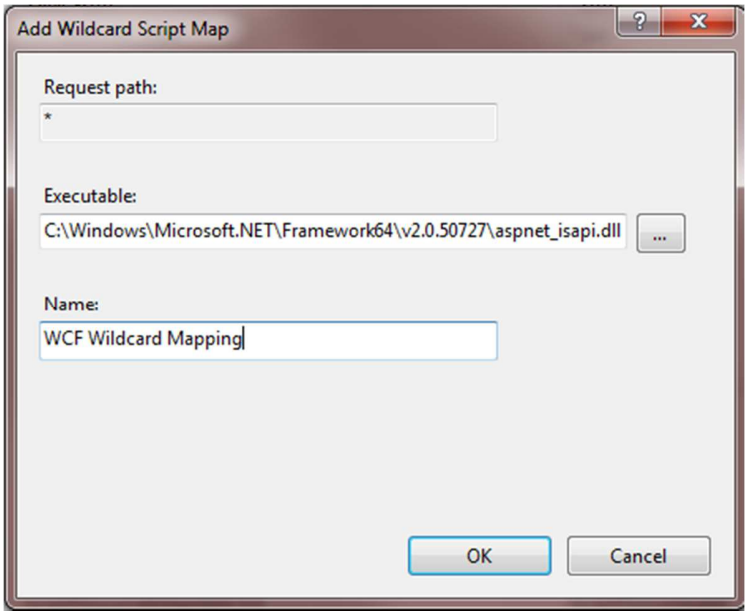
5) Create a second virtual directory under the default website root called ProfilesSPARQLAPI and map its physical location to the drive and directory that will host the ProfilesSPARQLAPI physical web files and resources.
    - This step can be setup as a standalone website or sub web of existing website. 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.
6) Create a third virtual directory under the default website root called ProfilesSearchAPI and map its physical location to the drive and directory that will host the ProfilesSearchAPI physical web files and resources
    - 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.
7) Optional: Perform a release build of the Profiles Solution in the Visual Studio 2008 IDE. This step can be avoided by using the pre compiled binary files included in the release download.
    -
8) Copy the binary files, or publish the Profiles project files into its physical location for hosting.
9) Copy the binary files, or publish the ProfilesSPARQLAPI project files into its physical location for hosting.
10) Copy the binary files, or publish the ProfilesSearchAPI project files into its physical location for hosting.
11) 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_WPG local server user account with read access to the same location.
12) From the web server file explorer, navigate to the physical location of the ProfilesSPARQLAPI web service hosted files and provide the [ServerName]/IUSR account read access.
    - For IIS6.0 provide the [ServerName]/IIS_WPG local server user account with read access to the same location.
    - For IIS7.0 provide the [ServerName]/IIS_IUSRS local server user account with read access to the same location.

13) From the web server file explorer, navigate to the physical location of the ProfilesSearchAPI web service hosted files and provide the [ServerName]/IUSR account read access.

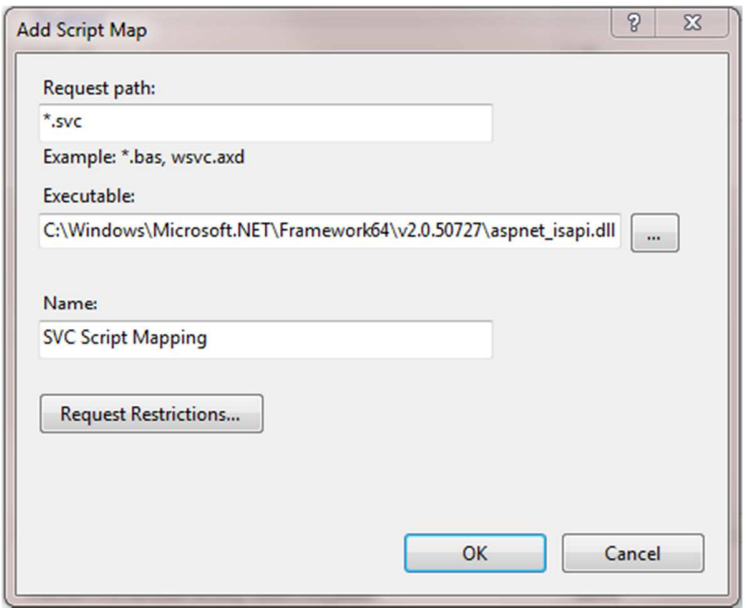
- For IIS6.0 provide the [ServerName]/IIS_WPG local server user account with read access to the same location.
- For IIS7.0 provide the [ServerName]/IIS_IUSRS local server user account with read access to the same location.

14) **If you are hosting Profiles on IIS7.x**, you will need to add Handler Mappings by using the aspnet_isapi.dll and setup your Application Pool for Classic Pipeline Mode.

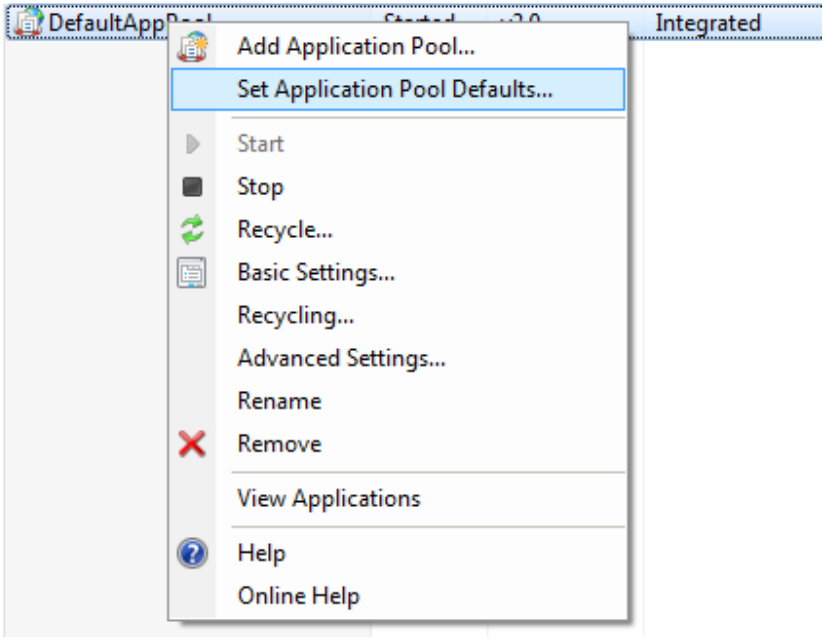
- For the website, you will need to add a Handler Mapping so that IIS will recognize RESTful URLs.
    1. Double click on the Handler Mapping icon for the Profiles application.
    2. Right click in the screen that displays the list of all current Mappings and select “Add Wildcard Map”.
    3. In the dialog box…
        a. For the Request path, enter “*.svc”.
        b. The aspnet_isapi.dll executable you map to will depend on if you are installing on a 64 or 32 bit App Pool. On a 32-bit enabled App Pool, the path will be “\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll”. On a 64-bit App Pool (32 bit disabled), the path will be “\WINDOWS\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll”. Leave “Verify that file exists” unchecked when selecting the file.
        c. For the Name, enter “Add Wildcard Maping”.
        d. Click “OK”.



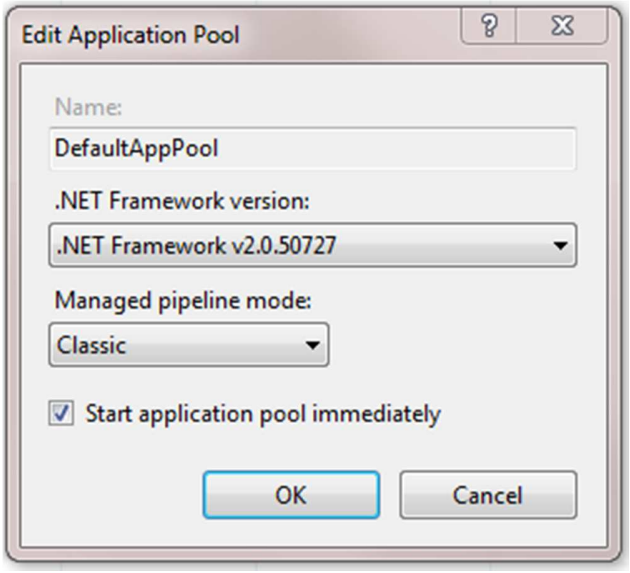
- For the APIs, you will need to add a Handler Mapping so that IIS will recognize *.svc files.
  1. Double click on the Handler Mapping icon for the Profiles application.
  2. Right click in the screen that displays the list of all current Mappings and select “Add Script Map”.
  3. In the dialog box…
     a. For the Request path, enter “*.svc”.
     b. The aspnet_isapi.dll executable you map to will depend on if you are installing on a 64 or 32 bit App Pool. On a 32-bit enabled App Pool, the path will be “\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll”. On a 64-bit App Pool (32 bit disabled), the path will be “\WINDOWS\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll”. Leave “Verify that file exists” unchecked when selecting the file.
     c. For the Name, enter “SVC Script Mapping”.
     d. Click “OK”.



- Setup your Application Pool for Classic Pipeline Mode:
  1. Open the application pool mapped to your Profiles Application.
  2. Right click and select Basic Settings…

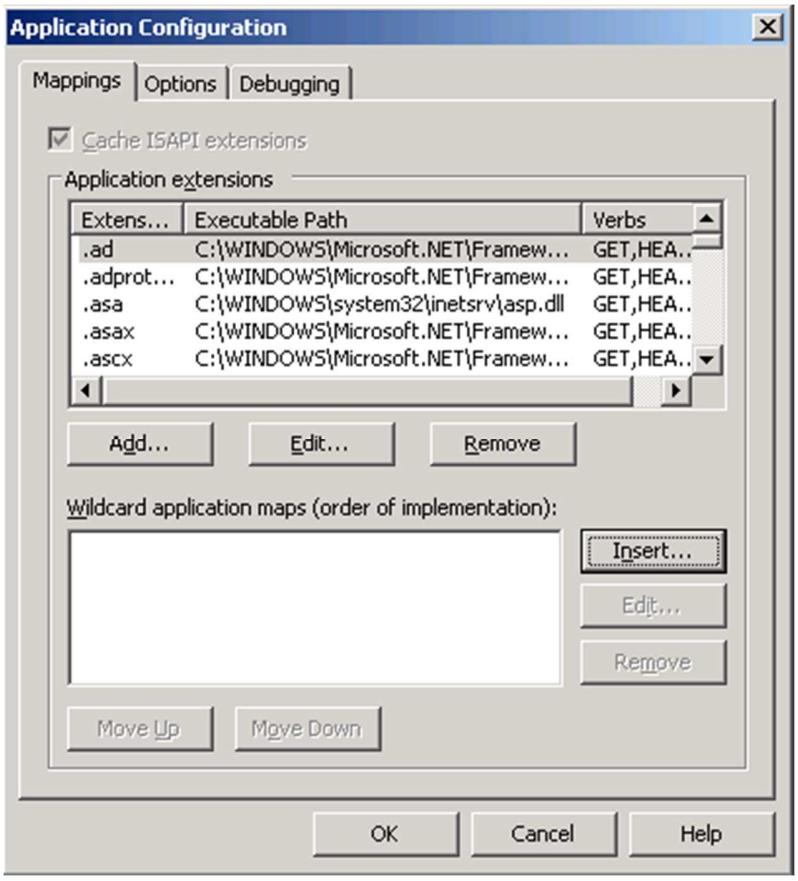


3. Select Classic from the Managed pipeline mode dropdown box, and click OK.

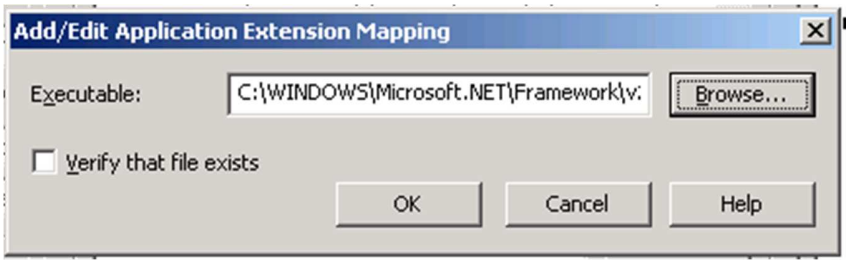


15) **If you are hosting Profiles on IIS6.x**, you will need to setup a wildcard mapping using the aspnet_isapi.dll. This will enable IIS to recognize the RESTful URL style of the Profiles framework requests.

   a) Open the properties dialogue of the Profiles virtual directory/application and select the Application Configuration button.



b) On the Mappings tab, click the “Insert…” button next to the “Wildcard application maps” area. Select the file aspnet_isapi.dll. On a 32-bit OS, the path will be “\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll”. On a 64-bit OS, the path will be “\WINDOWS\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll”. Leave “Verify that file exists” unchecked and then press OK on both dialog boxes.



16) **If you are hosting Profiles on IIS8.x**, you should follow the instructions for IIS7.x above.

- When creating the Wildcard Mapping on IIS8.x ensure that the Path Type is set to unspecified. Some installations have reported this defaulting to File. This causes RESTful URLs to return a 404 error.

17) Edit the Profiles web.config file for the following items:

- `connectionStrings/@name=ProfilesDB`: Provide the server name, database name, userid and password for the default connection string. Or you can customize your own security model and authentication process and integrated it into profiles .Net framework.
- `appSettings/@key=SPARQLEndPoint`: Set this to the full absolute URL of the SPARQL API (http://[yourdomain]/ProfilesSPARQLAPI.svc/Search).
- `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=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=aspnet:MaxHttpCollectionKeys`: This is set to 10000 by default to enable the display of large tables in Profiles RNS.

18) Edit the ProfilesSearchAPI web.config file for the following items:

- `connectionStrings/@name=ProfilesDB`: Provide the server name, database name, userid and password for the default connection string. Or you can customize your own security model and authentication process and integrated it into profiles .Net framework.
- `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.

19) Edit the ProfilesSPARQLAPI web.config file for the following items:

- `connectionStrings/@name=ProfilesDB`: Provide the server name, database name, userid and password for the default connection string. Or you can customize your own security model and authentication process and integrated it into profiles .Net framework.
- `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.

20) 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.

21) Optional: 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". Note that http://maps.google.com has a new Google Labs feature that shows you the latitude and longitude of where your mouse is pointing. This can be helpful for editing the config.xml file. To use this, go to http://maps.google.com/maps?showlabs=1 and enable the "LatLng Tooltip" option.

22) Optional: 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.

23) 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.

Once the web.config files are edited correctly, browse to the home page of profiles. If the error page displays in place of the home page, review the server event log for any details that will need to be addressed before proceeding.

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.

Installing the ORNG OpenSocial Extension (Optional)

OpenSocial will allow you to create new features in Profiles as independent applications that you “plug” into Profiles. The OpenSocial applications can be shared with other institutions, a number of these have been built by UCSF, Wake Forest and Baylor. A few of the ones from UCSF have been included in this version of Profiles.

ORNG stands for Open Research Networking Gadgets, and is based on merging the OpenSocial API with the Linked Data standard that is now supported by Profiles. If you install the OpenSocial extension, you will also enable Profiles to produce JSON-LD, an emerging standard for serializing RDF as JSON.

It is strongly recommended that you first install and test the base Profiles product before attempting to install and configure OpenSocial. The OpenSocial extension requires a Java/Tomcat installation and some extended IIS configuration, all of which is outlined in the ORNG documentation. Because this is an optional component, the installation and support documents are located in a separate ORNG folder. Please start with the ORNGInstallation_Guide.

Note that the ORNG extension can be disabled at any time by setting enabled to false in the ORNG section of the Web.config.