Profiles Research Networking Software (Beta Version) Installation Guide

Documentation Version: July 22, 2011

Software Version: July 22, 2011 (ProfilesRNSBeta110722)

Table of Contents

<u>Hardware and Operating System Requirements</u>

Profiles is a Microsoft .NET 3.5 website that uses a Microsoft SQL Server 2005 (or 2008) database. 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 10 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.

Required software packages:

- Microsoft .NET 3.5

- IIS6 or IIS7.X

- Microsoft SQL Server 2005 or 2008 with the following components installed:

    o SQL Server Agent

    o SQL Server Integration Services

    o All current service packs (including service pack 3 if using SQL Server 2008).

- Microsoft Visual Studio 2008

Outline of Installation Process

1. Install the database

2. Place basic HR data (name, email, address, affiliation, faculty rank, etc.) into database "loading" tables.

3. Run a series of processes that use the loading tables to populate the remaining database tables and automatically locate publications from PubMed.

4. Setup scheduled database jobs to keep the data up-to-date.

5. Install and configure the website code.

Database Components

There are three database components of Profiles:

1. There is a profiles database that contains the data tables, stored procedures, and other database objects used by the website. It is distributed as a backup file, which you can restore into an empty SQL Server database. Some of the tables in the backup file are pre-populated with data, such as a set of tables that contain the Medical Subject Heading (MeSH 2010) vocabulary. You will need to provide the database with basic person information (e.g., names, titles, and affiliations), typically as a one-time data load or an automated feed from your human resource (HR) database.

2. There are three SSIS packages that are used to pull data from external sources. The first takes the person data you loaded into Profiles and calls a "disambiguation engine" web service hosted at Harvard to obtain Medline/Pubmed article IDs for those people. The second retrieves the full citation data for those Pubmed IDs. The third takes address information you provided about people and calls a Google API to convert these to latitude and longitude coordinates.

3. To optimize performance, there are several "cache" tables in the database, which contain aggregate counts and denormalized copies of the raw data. To keep these cache tables up-to-date, Profiles contains a series of database jobs, which can be scheduled to run periodically to rebuild these tables.

Installing the Database

Follow the steps below to install the database:

1. Unzip the compressed database file.
2. Restore Profiles_Core.bak to an empty SQL Server database.

   - The recommended database name is "Profiles"

   - Make sure SQL Server, SQL Server Agent and SQL Server Integration Services are up and running

   You can check a service's status from **Start → Control Panel → Administrative Tools → Services**
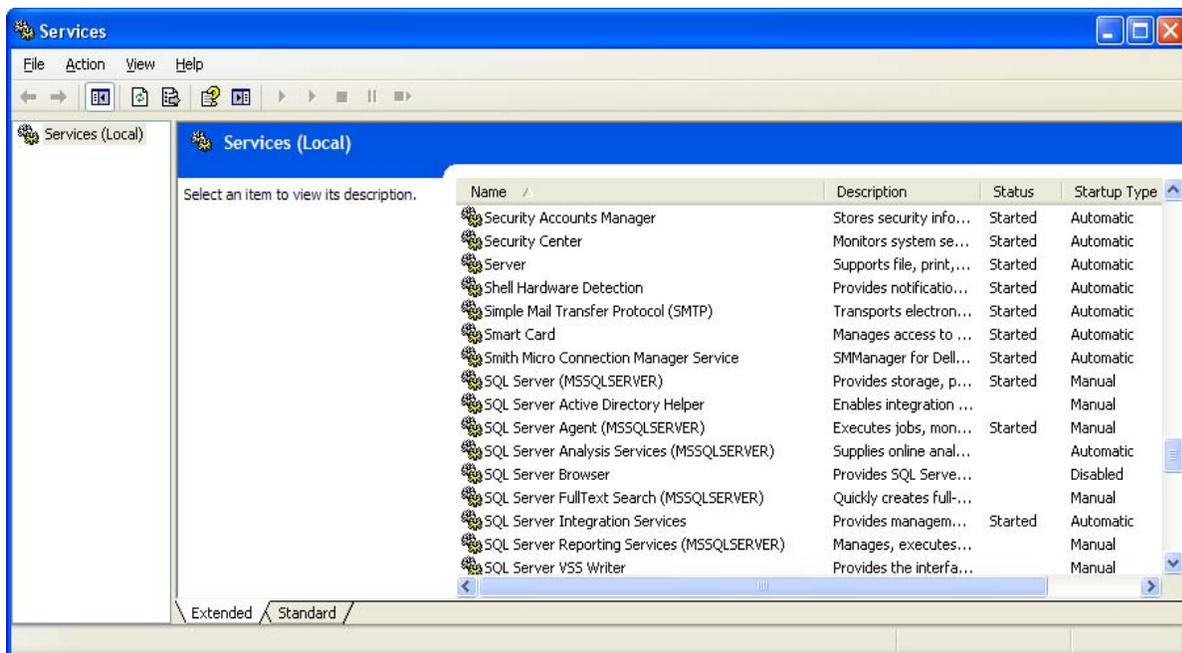
   - Launch Microsft SQL Server Management Studio

   Start → All Programs → Microsft SQL Server 2005 → SQL Server Management Studio and then connect to your SQL Server using either Windows Authentication or SQL Server Authentication. Once you connected, you will see a screen like the following:
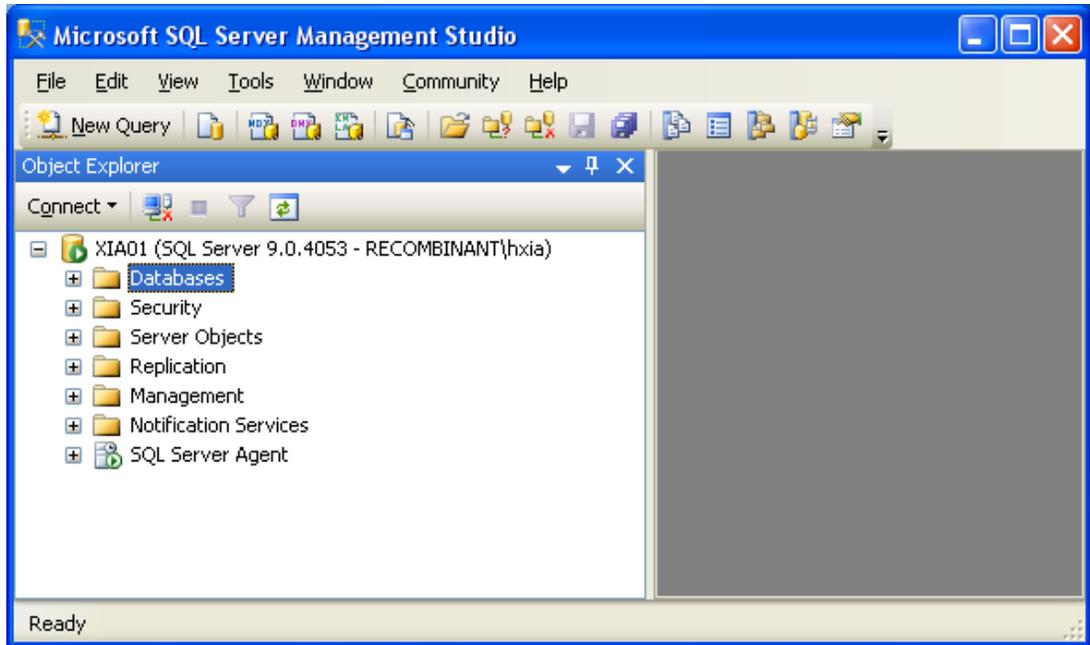
In the above screen shot, you can see there is a green arrow close to SQL Server Agent, which means the service is up and running.

- Restore database

Right click **Databases** → Choose **Restore Database …** and

1. Set **To database** to Profiles
2. Check **From device** radio button and pick up the file from your filesystem
3. Check **Restore** checkbox
4. Enter **OK**

The database backup for profile is loaded into your SQL Server as **Profiles**. If you click **Databases** node in your Microsoft SQL Server Management Studio, you should be able to see **Profiles** from the displayed list.
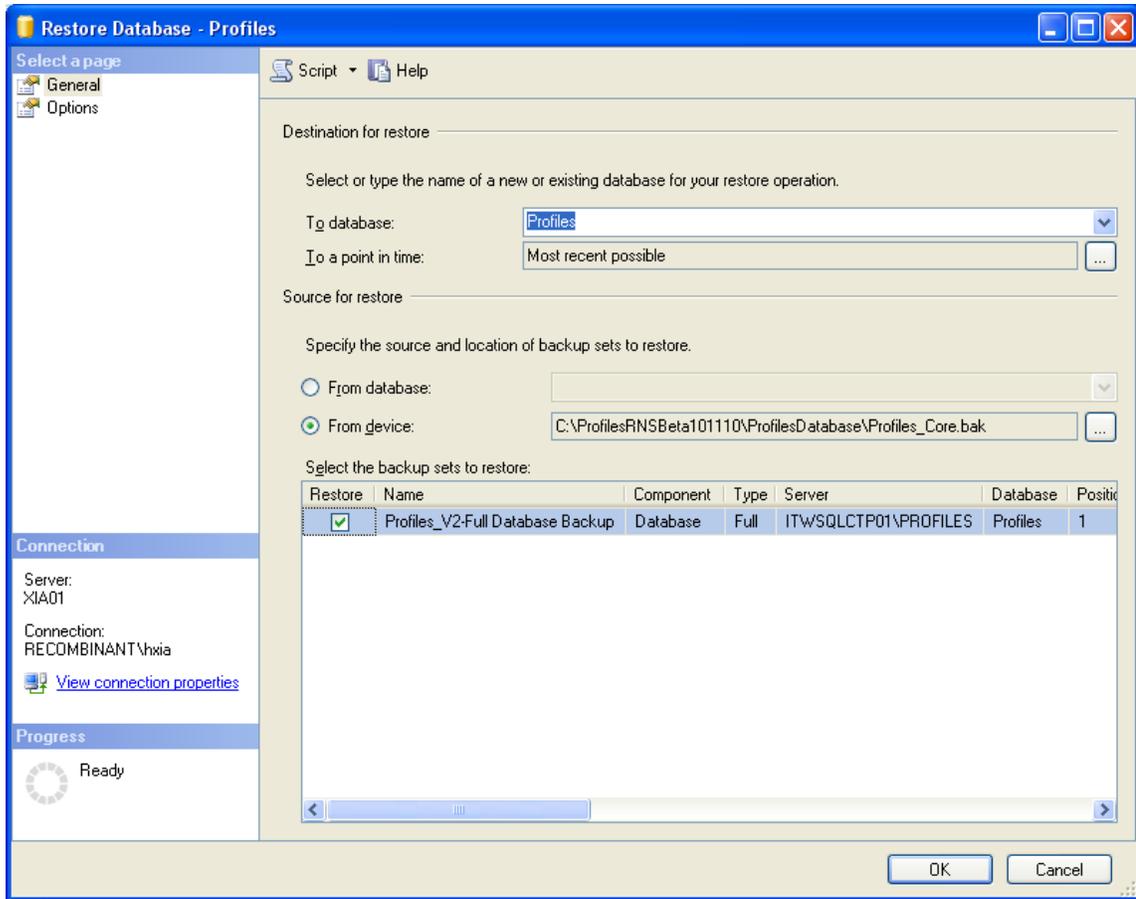
3. Import the following SSIS packages into the SQL Server msdb database:

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.

- Left click **Connect** (the left corner of the Studio)
- Pick **Integration Services…**

  You will notice an Integration Services node added to the left panel.
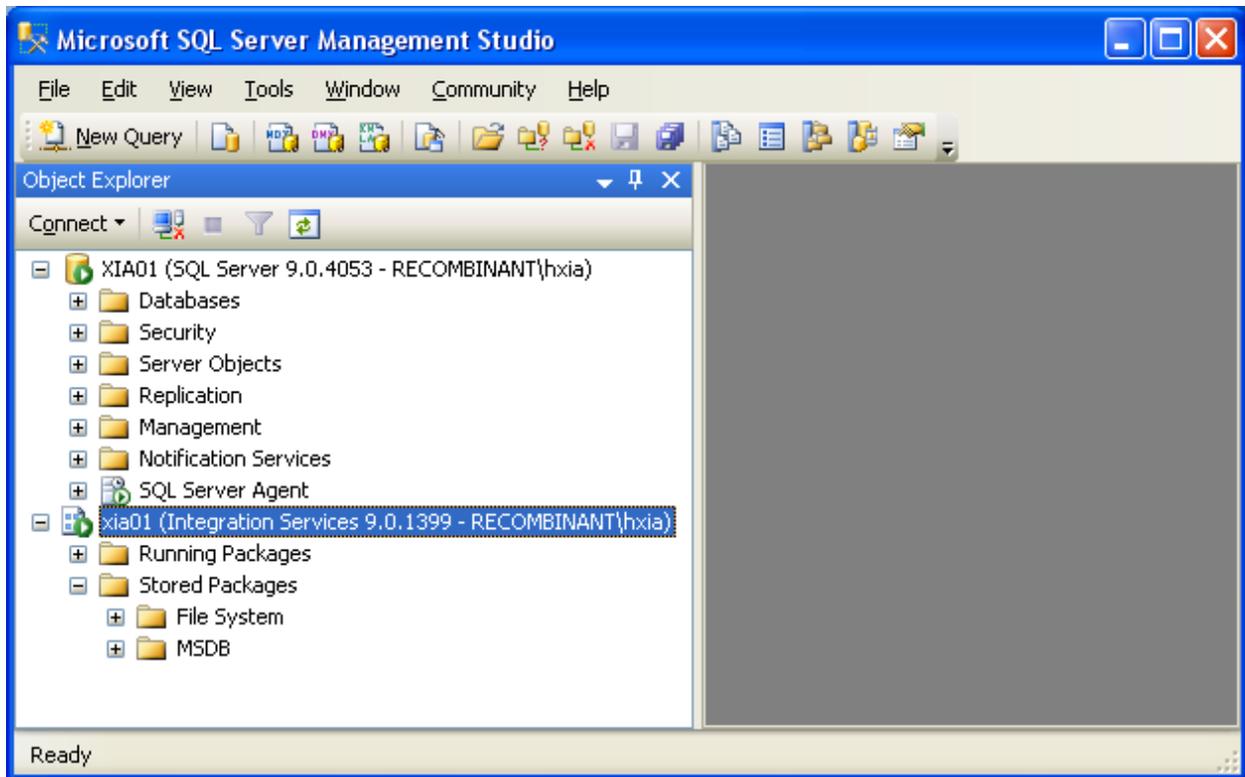
- Expand newly added **Integration Services** node and you'll see **Running Packages** and **Stored Packages** nodes

- 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:

  1. **Package location**: **File System**

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

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

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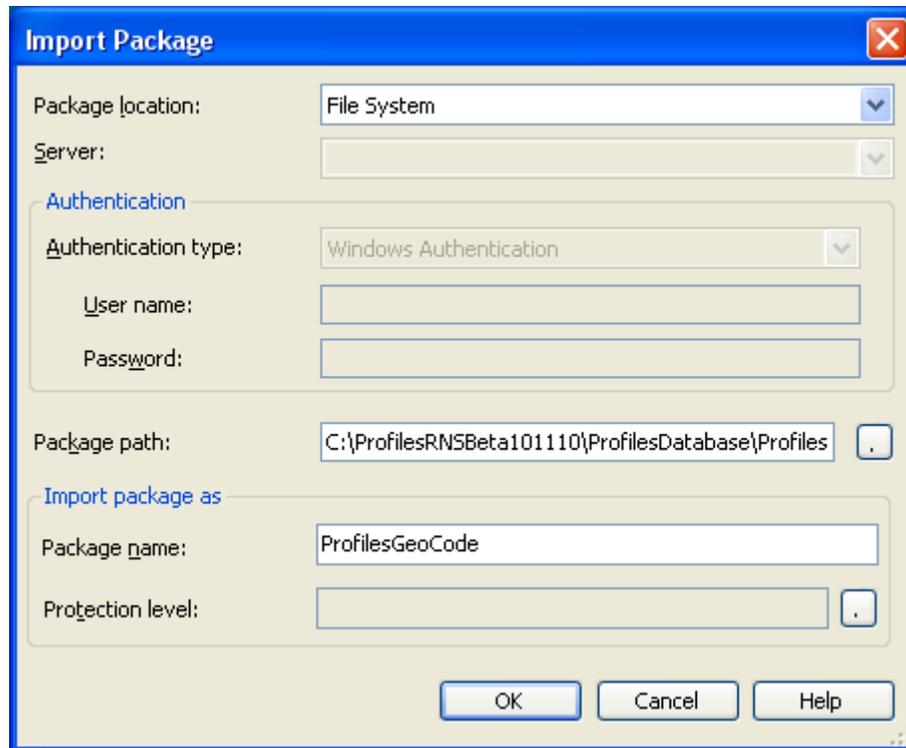
You use this procedure to install the following three packages. Note that there are different versions of these packages for SQL Server 2005 and SQL Server 2008. The 2005 packages are in a folder named SQL2005, and the 2008 packages are in a folder named SQL2008.

1. PubMedDisambiguation_GetPubs.dtsx

2. PubMedDisambiguation_GetPubMEDXML.dtsx

3. ProfilesGeoCode.dtsx

Successfully installed packages will be displayed under MSDB node as the following:
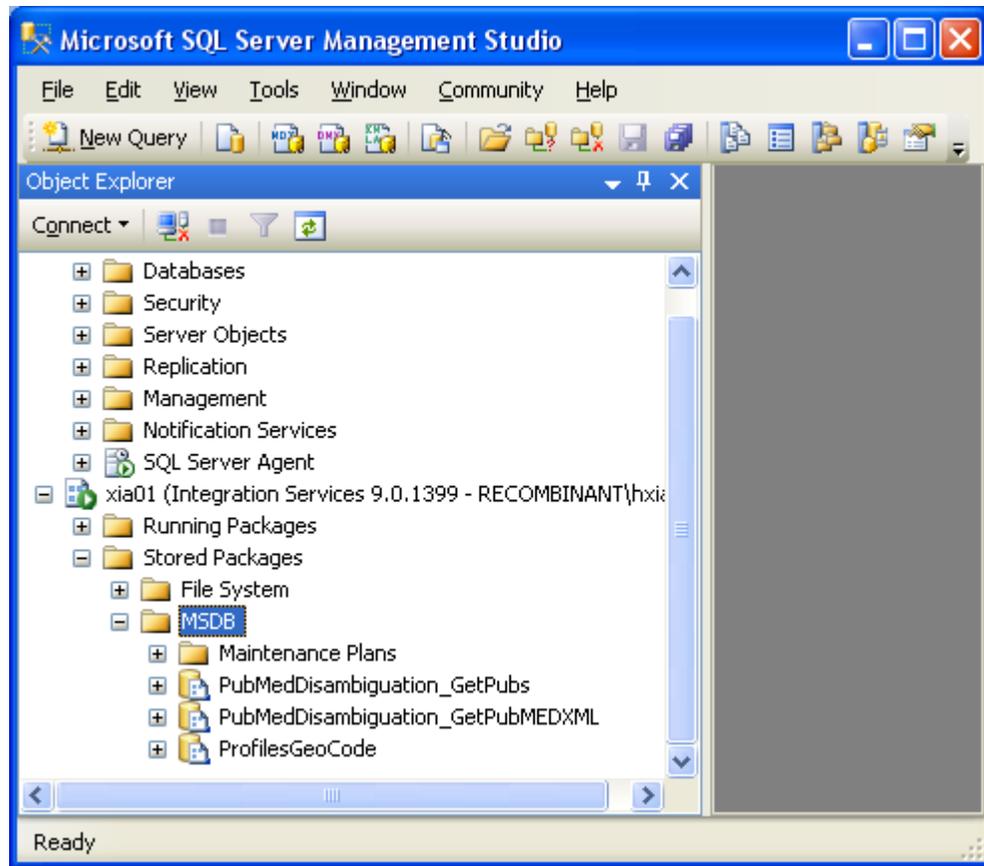
4. Create an admin account with sysadmin privileges that can be used to execute a series of scheduled jobs.

  - The script "ProfilesSetUserPermissions.sql" will create a default admin/application user account, AppProfilesUser, with all of the appropriate permissions.

  - If you want to use a database user account other than AppProfilesUser, then that account will need to be a sysadmin and you will have to grant execute permissions for all stored procedures in the Profiles database. (We recommend that you use the ProfilesSetUserPermissions.sql script as a template to help you do this. Just do a global search and replace of the AppProfilesUser, with your new database username. The default password can also be changed in the first part of this script.

    NOTE - AppProfilesUser is the recommended/default account.)

  - If you want to use a database user account other than the default account setup provided in the Profiles_Core.bak database, then remember to grant appropriate stored procedure execute permissions. The script ProfilesSetUserPermissions.sql is a template to help you do this. Just do a global search and replace of the AppProfilesUser, with your new database username.

The default password can also be changed in the first part of this script. NOTE - AppProfilesUser is the recommended/default account.

Once the user created, double check its permission and make sure it has both **public** and **sysadmin** persmissions using the studio: **Security → Logins →** right click <User created> → **Properties → Server Roles**.

If the sysadmin role cannot assign to this newly created user, you need to make sure this user can install SSIS packages and run jobs.

5. The following scripts that create scheduled jobs need to be modified so that they work in your particular environment:
   - ProfilesYearlyJobs.sql
   - ProfilesNightlyJobs.sql
   - ProfilesWeeklyJobs.sql
   - ProfilesMonthlyJobs.sql
   - ProfilesHourlyJobs.sql
   - PubMedDisambiguation_GetPubs.sql
   - PubMedDisambiguation_GetPubMEDXML.sql
   - ProfilesGeoCodeJob.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 two versions of PubMedDisambiguation_GetPubs.sql and PubMedDisambiguation_GetPubMEDXML.sql, depending on whether you are using SQL Server 2005 or 2008. The 2005 files are in the folder SQL2005, and the 2008 files are in the folder SQL2008.

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

7. Execute the scripts you modified in steps #5 and #6.

Loading Person Data

Profiles requires that you provide basic demographic data about people. This can be a one-time import, or more likely you will want to setup a data feed from a human resources (HR) system that updates Profiles nightly. The general process is that you place the data in a set of "loading" 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 loading tables:

   a. The _person table has <u>one row per person</u> and includes fields such as first name, middle name, last name, name suffix, email, phone, fax, and address.

   b. The _person_affiliations table lists a person's titles, institutions, departments, divisions, and faculty rank (e.g., "associate professor"). This table can have <u>multiple rows per person</u>, reflecting the multiple jobs or roles a person has in your organization.

   c. The _person_filter_flags 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 <u>multiple flags per person</u>. Flags can be grouped into categories. For example, "faculty", "staff", and "student" can be grouped into a category named "job type".

   d. The _user table has <u>one row per user</u> and includes basic user name and affiliation information. The _user table is used to create accounts for individuals who need access to the website, but will <u>not</u> have their own profiles. The same individual should not be listed in both the _person and _user tables.

3. The Profiles code has gone through multiple versions. In order to minimize the impact of changes on custom data feeds that some institutions have created for older versions of the software, we have chosen not to remove columns from the loading tables even if they are <u>no longer used</u> by Profiles. These defunct columns are noted in the column definitions below. These columns might be deleted in the future.

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

5. There is an "all-or-nothing" approach in the loading 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.

6. Each of the loading 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 loading 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.

7. Do not make changes to the data in the actual tables used by Profiles. Instead, always place corrections or updates in the loading tables, and re-run the loading 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.

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

9. We recognize that raw HR data is often "messy" and the institution, department, and division names stored in your HR database might be different than how you want these displayed on the Profiles website. In the _person_affiliations tables, place the actual names in the "institutionfullname", "departmentfullname", and "divisionfullname" columns. Indicate the names that should be displayed on the website in the columns "institution", "department", and "division". Even if they are the same value, you must provide both an actual name and a display name when listing institutions, departments, or divisions.

Below are the column definitions for each of the loading tables. The Data Type and Load Length describe the columns in the loading tables. The Max Length corresponds to the columns into which the data is copied during the loading process. Despite the size of the columns in the loading tables, if their length exceeds the Max Length, then the loading process might fail. Also, note that some columns, such as _person.floor are type nvarchar in the loading table, but need to have numeric values for the loading process to work.

1. Table: _person

Column	Data Type	Load Length	Max Length	Category
internalusername	nvarchar	2000	50	Required
internalldapusername	nvarchar	2000	50	Not Used
internalfacultyid	nvarchar	200	int	Not Used
internaljobcode	nvarchar	200	50	Not Used
internaldepttype	nvarchar	200	50	Not Used
numpublications	int		int	Not Used
firstname	nvarchar	2000	50	Optional

middlename	nvarchar	2000	50	Optional
lastname	nvarchar	2000	50	Required
displayname	nvarchar	2000	255	Required
degree	nvarchar	2000	50	Not Used
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	Not Used
city	varchar	1000	100	Not Used
zip	varchar	1000	10	Not Used
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
internalpersontype	varchar	1000	50	Not Used
title	nvarchar	2000	255	Not Used
internaladmintitle	nvarchar	2000	200	Not Used
emailaddr	nvarchar	2000	255	Optional
internalfullparttime	nvarchar	2000	50	Not Used
assistantname	nvarchar	2000	255	Not Used
assistantphone	nvarchar	2000	35	Not Used
AssistantEmail	nvarchar	2000	255	Not Used
AssistantAddress	nvarchar	2000	1000	Not Used
AssistantFax	nvarchar	2000	25	Not Used
AssistantUserID	nvarchar	2000	int	Not Used
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 column is 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 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.
e. 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 _user) table to isactive=0.
f. 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: _person_affiliations

Column	Data Type	Load Length	Max Length	Category
internalusername	nvarchar	2000	50	Required
internalldapusername	nvarchar	2000	50	Not Used
title	nvarchar	2000	200	Optional
internaladmintitle	nvarchar	2000	200	Not Used
emailaddr	nvarchar	2000	200	Don't Use
primaryaffiliation	bit		bit	Required
affiliationorder	tinyint		int	Required
institutionname	nvarchar	2000	500	Optional
institutionfullname	nvarchar	2000	500	Optional
institutionabbreviation	nvarchar	2000	50	Optional
departmentname	nvarchar	2000	500	Optional
departmentfullname	nvarchar	2000	500	Optional
departmentabbreviation	nvarchar	2000	50	Not Used
departmentvisible	bit		bit	Optional
divisionname	nvarchar	2000	500	Optional
divisionfullname	nvarchar	2000	500	Optional
divisionabbreviation	nvarchar	2000	50	Not Used
facultyrankcode	nvarchar	200	50	Don't Use
facultyrank	varchar	1000	100	Optional
facultyrankfullname	varchar	1000	100	Not Used
facultyrankorder	tinyint		tinyint	Optional
internalfullparttime	nvarchar	2000	50	Not Used
is_visiting	bit		bit	Not Used

Notes:

a. Each person must have exactly one row in _person_affiliations 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 and facultyrankcode columns are not used by Profiles; however, you must set these columns to NULL for the loading process to work properly.

    f. Facultyrankorder is <u>required</u> if you are using the facultyrank column. Every distinct facultyrank value in the _person_affiliations 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: _person_filter_flags

Column	Data Type	Load Length	Max Length	Category
internalusername	varchar	50	50	Required
filterrank	int		int	Not Used
personfilter	varchar	50	50	Required

Notes:

    a. After running the data load process, the person_filters table will be populated with a distinct list of personfilter values from the _person_filter_flags table. The person_filters.PersonFilterCategory and person_filters.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: _user

Column	Data Type	Load Length	Max Length	Category
internalusername	nvarchar	2000	50	Required
internalldapusername	nvarchar	2000	50	Not Used
firstname	nvarchar	2000	100	Optional
lastname	nvarchar	2000	100	Required
displayname	nvarchar	2000	255	Required
phone	nvarchar	2000	35	Not Used
emailaddr	nvarchar	2000	255	Not Used
institutionfullname	nvarchar	2000	500	Optional
departmentfullname	nvarchar	2000	500	Optional
canbeproxy	bit		bit	Required

Notes:

    a. Only list individuals in this table who are not listed in the _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 loading tables. Double check that you have <u>not</u> done any of these common errors:

1. ERROR: There are values in the loading tables that are longer than the allowed Max Length.
2. ERROR: The same internalusername value is being used more than once in the _person or _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 _person_affiliations.primaryaffilation is set to NULL.
7. ERROR: There is more than one record per person in _person_affiliations with primaryaffilation=1.
8. ERROR: A person does not have any records in _person_affiliations with primaryaffiliation=1.
9. ERROR: The required column _person_affiliations.affiliationorder is set to NULL.
10. ERROR: The same _person_affiliations.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 loading tables, confirm that you have not made any of the above errors by executing the stored procedure "**usp_ValidateProfilesLoaderTables**". 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 "**usp_loadprofilesdata**". This will parse the data in the loading 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 _person and _user tables as the PersonID and UserID. Otherwise, Profiles will generate its own unique values using sequential integers.

User Passwords

The data load process does not generate passwords for users, and there is no administrative interface to setup accounts. You must manually enter passwords into the user.password column or build your own data feed to populate this column. Many institutions replace the default authentication C# code within the Profiles RNS application with custom code that uses the institution's existing authentication mechanisms. See the section below on Website Authentication Options for more information.

Geocoding and Google Maps

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 _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. Register for a Google Map Key (http://code.google.com/apis/maps/signup.html). Google Map Keys are unique to the domain of the web page rendering the map. So, you might need to register multiple Google Map Keys, depending on how you setup Profiles in your environment. After you register the keys, insert them into the Profiles database table named "google_keys". You will need to create two records for each key. For one record, the google_keys.app column should be set to "maps", and for the other record, the google_keys.app column should be set to "geocode". In the google_keys.domain column, place the domain you used to register the Google Map Key. In the google_keys.gkey column, place the sequence of characters Google gave you when you registered for the key. See the example below:

```
insert into google_keys (app, domain, gkey)
   values ('maps','http://yourprofilesdomain','ABQIAAA…7BVmuODxkQ')

insert into google_keys (app, domain, gkey)
   values ('geocode','http://yourprofilesdomain','ABQIAAA…7BVmuODxkQ')
```

3. After you have completed loading the person data, run the ProfilesGeoCode job. 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.

4. In the website web.config file, set the default latitude and longitude when maps are displayed in Profiles. Do this by modifying the values for the keys GoogleMapCenterLatitude and GoogleMapCenterLongitude.

5. Optional: When displaying Google Maps, Profiles lets the user select from several pre-defined zoom 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 configurations in the google_map_preferences table. One row in this table should be flagged as DefaultLevel = 1. 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 google_map_preferences table.

In both steps 4 and 5, you will need to know the latitude and longitude of your institution. Note that the Google Maps website has a new "Google Labs" feature that displays latitude and longitude. Go to http://maps.google.com/maps?showlabs=1 and enable the "LatLng Tooltip" option to use this feature to figure out your latitude and longitude.

__Obtaining Publications__

In order for Profiles to automatically locate publications for people, you first need to provide a list
of affiliation strings in the disambiguation_pm_affiliations 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 disambiguation_pm_affiliations (affiliation)
   values ('%Harvard Medical School%')
insert into disambiguation_pm_affiliations (affiliation)
   values ('%Beth Israel Deaconess Medical Center%')
insert into disambiguation_pm_affiliations (affiliation)
   values ('%BIDMC%')
insert into disambiguation_pm_affiliations (affiliation)
   values ('%@hms.harvard.edu%')
insert into disambiguation_pm_affiliations (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.
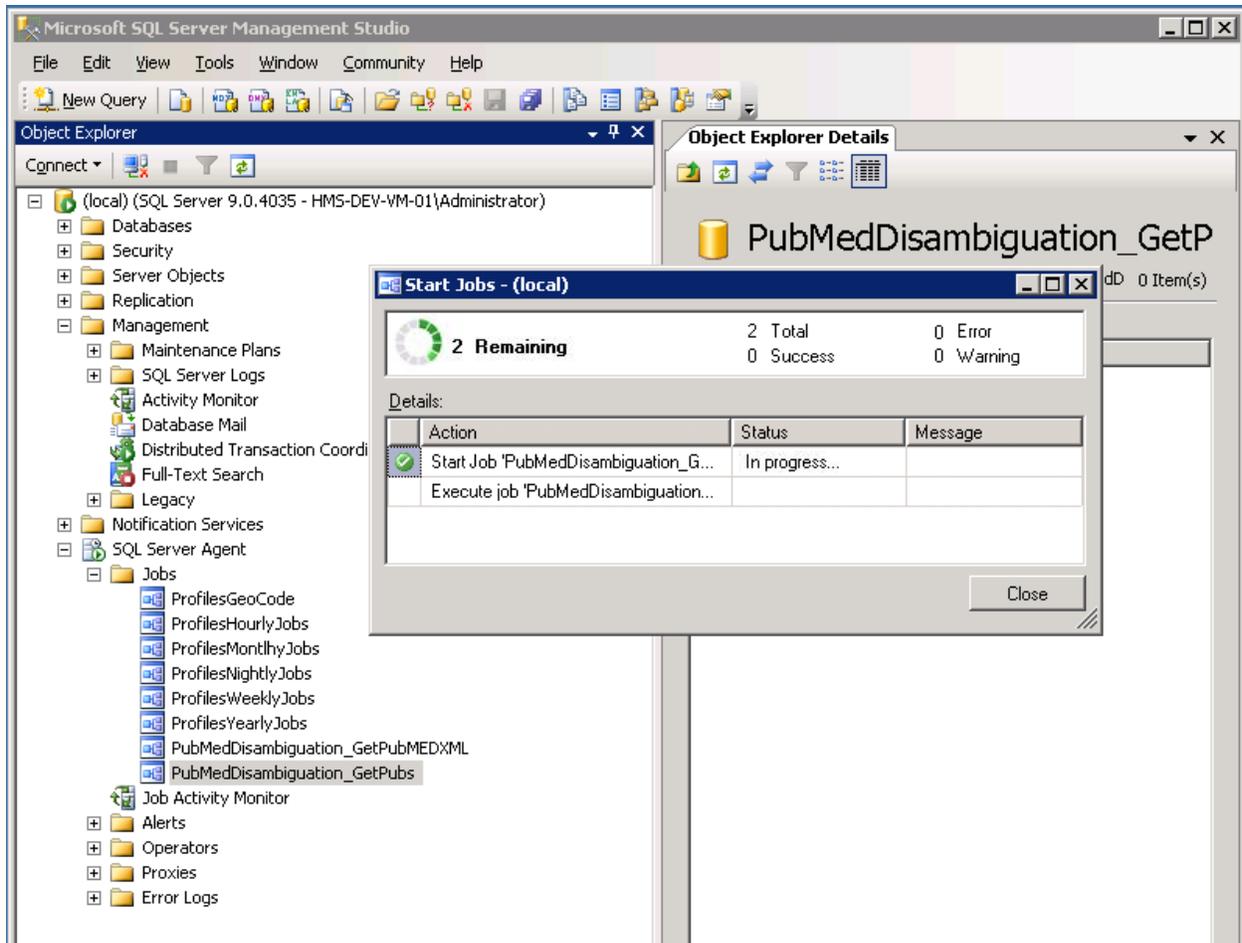
Once this job has completed (typically several hours), you should run the PubMedDisambiguation_GetPubMEDXML job. This job will retrieve the full xml for the pubmed articles and parse it in your local profiles instance.
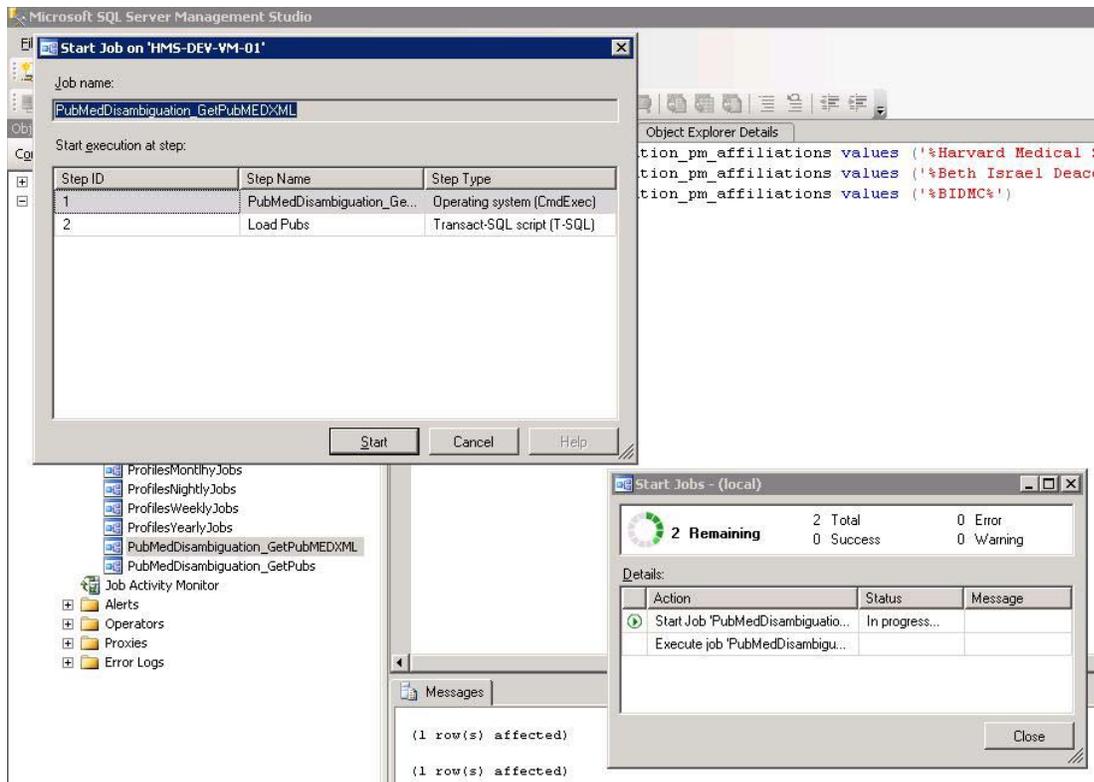
There are a few important technical notes about the service:

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. The URL of the service is likely to change in the next few months. If you are unable to connect to the service, please contact us for details.

3. 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 usp_GetPersonInfoForPubMed_xml 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 usp_GetPersonInfoForPubMed_xml 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 usp_GetPersonInfoForPubMed_xml 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.

Scheduling Database Jobs

Run ProfilesYearlyJobs.sql manually once after the initial install of Profiles. It will only need to be run again if you update the mesh_* tables. NLM publishes a new version of MeSH once a year.

The following jobs should be scheduled to run regularly:

- ProfilesNightlyJobs.sql
- ProfilesWeeklyJobs.sql
- ProfilesMonthlyJobs.sql
- ProfilesHourlyJobs.sql

The names indicate suggested run frequencies. However, the more often you run these better since they are required to maintain consistency between the content in profiles (e.g., publications) and other aspects of the website, such as the search results and passive networks. ProfilesWeeklyJobs.sql and ProfilesMonthlyJobs.sql should ideally be run whenever ProfilesNightlyJobs.sql is run. The weekly and monthly jobs, though, take significantly longer to run than the nightly job, so it might not be practical to run all of these nightly if your database server is not fast enough.
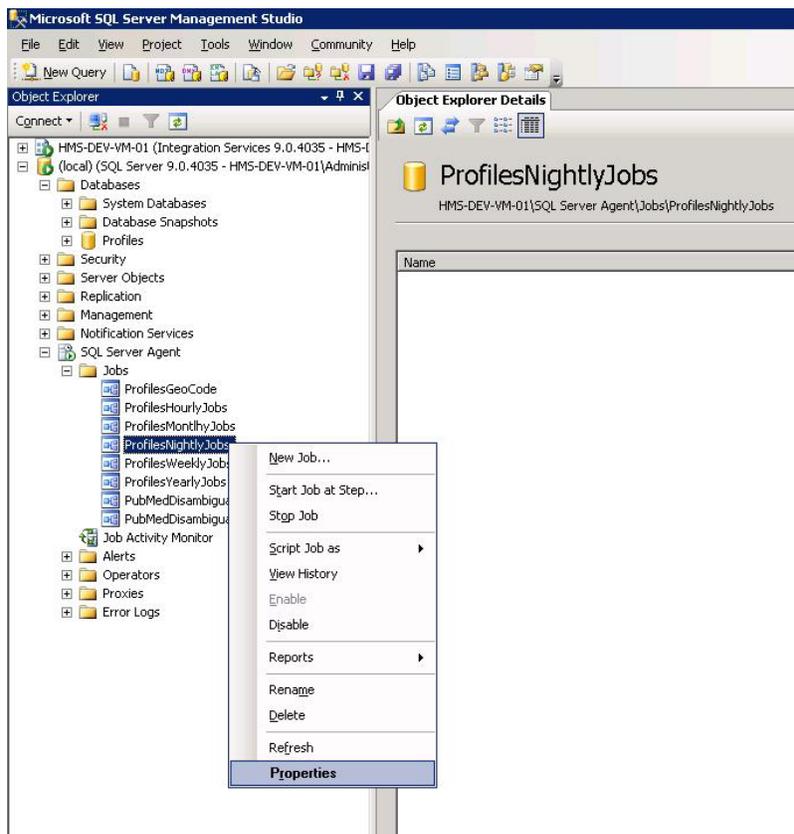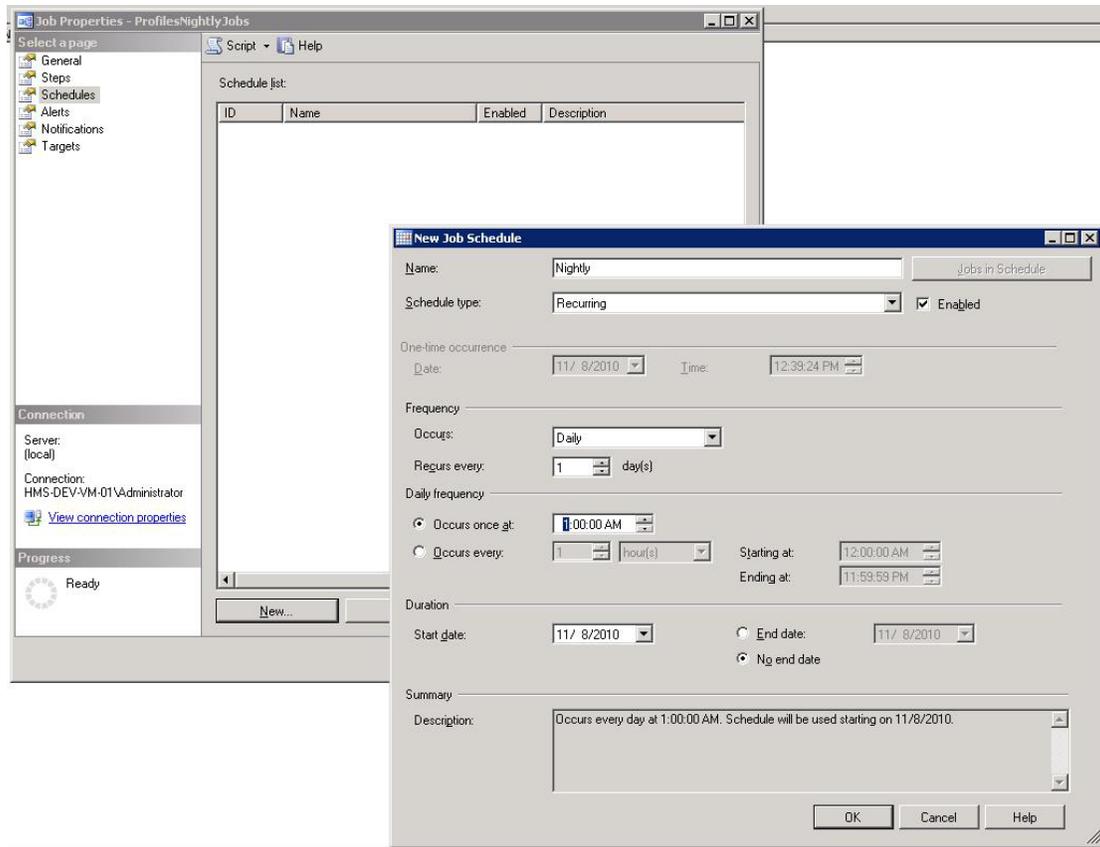
The ProfilesHourlyJobs.sql runs independently of the others. However, there are dependencies among the remaining jobs, and they should be run in this order: ProfilesNightlyJobs.sql → ProfilesWeeklyJobs.sql → ProfilesMonthlyJobs.sql. Also, if you plan to schedule usp_loadprofilesdata to update your person and user data (see below in the section on updating person data after go-live), then this should be done before ProfilesNightlyJobs.sql is run.

Installing the Website

The Profiles beta website contains two main components that are comprised of a shared layer of developed and Microsoft class libraries: The two main components are the actual website, and a separate application for the Profiles web services. The entire solution should contain the main website (ProfilesWeb) and the web service (Connects.Profiles.Service) with each project referencing the required class libraries.

Solution name:
      ResNavPeople

Class Libraries:
      (Developed)
      Connects.Profiles.Data
      Connects.Profiles.BusinessLogic
      Connects.Profiles.Common
      Connects.Profiles.Utility
      Connects.Profiles.Service.ServiceContracts
      Connects.Profiles.Service.DataContracts
      Connects.Profiles.ServiceImplementation

      (Microsoft)
      FUA (Ajax file uploader)
      Microsoft Enterprise Library

Website:
      ProfilesWeb

Web Service:
      Connects.Profiles.Service (API and Network Browser Service)

Installation steps:

Profiles is designed to run as two different web sites or as two virtual directories on a single server IIS instance or across two different server instances. If you are hosting the Profiles web application and Profiles web service on two 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. 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

   - This path might be different depending on a 32 vs 64 bit operation system.

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 ProfilesService and map its physical location to the drive and directory that will host the Connects.Profiles.Service physical web service 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

6. Publish or place the ProfilesWeb project files into its physical location for hosting.

7. Publish or place the Connects.Profiles.Service project files into its physical location for hosting.

8. From the web server file explorer, navigate to the physical location of the ProfilesWeb 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.

9. From the web server file explorer, navigate to the physical location of the Connects.Profiles.Service 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.

10. The Profiles web site and web service are designed to be hosted as a compiled application, or as raw source code files for JIT compile on activation of the first user.

    - For hosting as a compiled application you will need to load the solution into the IDE, and then perform a release build of the entire solution.

      1. Review all messages and outputs of the compiler to ensure all references to required class libraries are in place.

      2. Publish Connects.Profiles.Service to its target hosted location.

      3. Publish the ProfilesWeb project to its target hosted location.

    - For JIT Compile on activation of the first user.

1. Copy and paste the ProfilesWeb folder contents to your physical hosting location.

2. Copy and paste the Connects.Profiles.Service folder contents to your physical hosting location.

11. Edit the Connects.Profiles.Service web.config file for the correct connection string. Other settings should be left as default.

12. If you are hosting the Connects.Profiles.Service on IIS7.x you will need to add two Handler Mappings by using the aspnet_isapi.dll. This will enable IIS to recognize the .svc file extension and allow the RESTful URL style of the WCF request.

- Double click on the Handler Mapping icon for the Connects.Profiles.Service application.

- Right click in the screen that displays the list of all current Mappings and select "Add Script Map"

- The aspnet_isapi.dll executable you map to will depend on if you are installing on a 64 or 32 bit OS version.
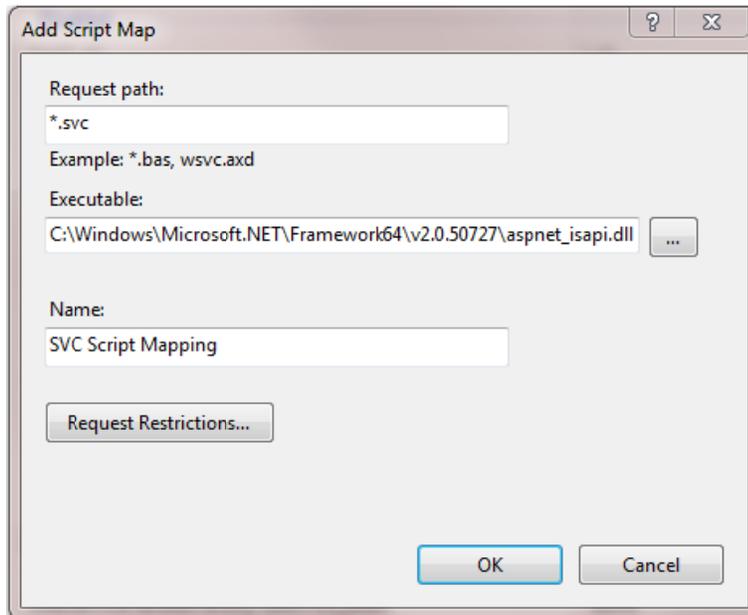
- After pressing OK, right click in the screen that displays the list of current Mappings and select "Add Wildcard Map".
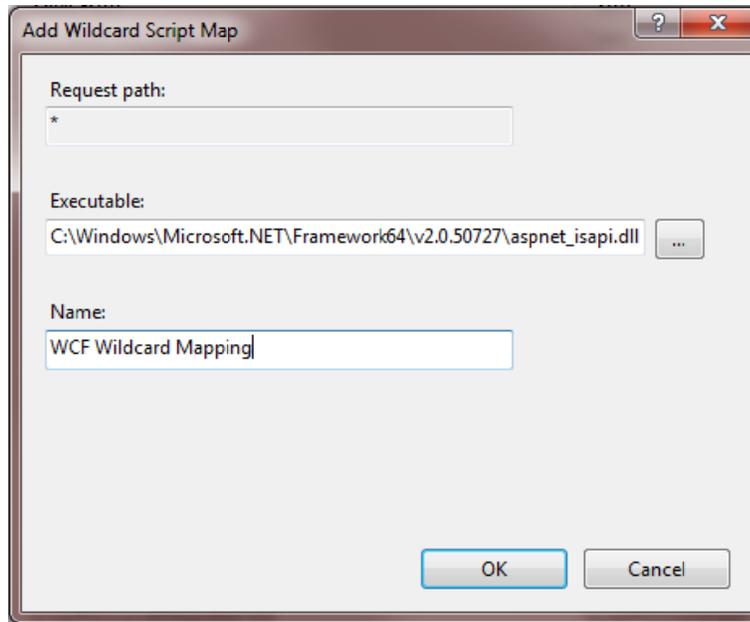
13. If you are installing Profiles on a 64-bit machine, you need to set IIS to enable 32-bit application. (The membership/user framework used by Profiles has a dependency on the 32-bit version of System.Data.SQLite.)

IIS7:
- Go to Application Pools->Profiles->Advanced Settings and set Enable 32-bit applications = True

In IIS6:
- Click Start, click Run, type cmd, and then click OK.
- Type the following command to enable the 32-bit mode:
  ```
  cscript %SYSTEMDRIVE%\inetpub\adminscripts\adsutil.vbs
  SET W3SVC/AppPools/Enable32bitAppOnWin64 1
  ```
- Type the following command to install the version of ASP.NET 2.0 (32-bit) and to install the script maps at the IIS root and under:
  ```
  %SYSTEMROOT%\Microsoft.NET\Framework\v2.0.50727\aspnet_r
  egiis.exe –i
  ```
- Make sure that the status of ASP.NET version 2.0.50727 (32-bit) is set to Allowed in the Web service extension list in Internet Information Services Manager.

14. Edit the ProfilesWeb web.config file for the following items:

- Connection string: connectionStrings.ProfilesDB

- AppSettings.URLBase: needs to reflect the full URL of the profiles website to include a closing backslash.   Example: http://[Domain]/ProfilesWeb/

- AppSettings.HomepageUrl: needs to reflect the full URL of the default page for the profiles website. Example: http://[Domain]/ProfilesWeb/Home.aspx

- AppSettings.ProdHost: needs to be set as localhost.

28

- AppSettings.ConnectsLoginURL: login screen for the edit enabled profiles functions. Example: http://[Domain]/ProfilesWeb/Login.aspx

- AppSettings.AfterLogoutURL: redirect page you want your user to be taken to once they have logged out of profiles.

- AppSettings.DefaultPersonImageURL: the default image that will show in the event a photo of a person does not exist.

- AppSettings.ProfileResponseXSD: Path of xml schema if schema validation is required.

- AppSettings.GoogleMapCenterLatitude: used to center the default of all network maps.

- AppSettings.GoogleMapCenterLongitude: used to center the default of all network maps.

- AppSettings.HideInstitutionSelectionForSearch: used to hide the institution field from the search options.

- AppSettings.HideDepartmentSelectionFromMiniSearch: used to hide the department option from the mini search options.

- AppSettings.NetworkBrowserService: the full url of the deployed Connects.Profiles.Service Network Browser to include search method and closing backslash. Example: http://[Domain]/ProfileService/NetworkBrowserService.svc/profiles/

- RoleManager.Providers: change the application name to the full path of the Profiles application to include closing backslash. Example: http://[Domain]/ProfilesWeb/

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.

Website Authentication Options

The Profiles web site utilizes the membership provider model built into Internet Information Server (IIS) as a means for authentication that is flexible and extensible to many different environments.

By default, Profiles uses basic IIS forms authentication that has been modified to utilize the [user] table in the Profiles database. This functionality is encapsulated in a default provider called "ProfilesDBMembershipProvider" that is defined in a class found in the App_Code folder of the Profiles web site.

Profiles installations that wish to integrate their own existing authentication mechanism can simply build their own C# class that inherits from System.Web.Security.MembershipProvider and include a reference to this provider in the web.config on their site. The settings can be found in the <membership> section of the web.config.

When developing a provider class that inherits from a built-in class, you typically override those methods that you wish to customize and control. Each installation will vary, but in most cases the ValidateUser and FindUserByName methods will need overrides in the custom provider class.

In the case of installations where a Single Sign-On (SSO) solution is desired, the authentication is performed outside of Profiles and what is typically required is a mechanism for handling the "callback" from the SSO system back to Profiles. The requirement here is that the SSO system validate the user and on the return, the core ASP.NET Profile attributes be filled with data from the SSO system. Depending on the SSO system, this information may be available on the QueryString, HTTP header or a cookie.   The ASP.NET Profile attributes to be set can be found in the web.config in the <profile> section.

Modifying the Website's Look-And-Feel

The ProfilesWeb site is based on asp.net technology with a presentation layer comprised of .Net Master pages, HTML, CSS, .Net controls and .Net user controls.

- CSS: A folder called CSS exists under the ProfilesWeb project root. It contains 8 css files and one HTC support file for custom styles.
- Images: A folder called Images exists under the ProfilesWeb project root. This contains all image media content for the ProfilesWeb site. Many images are used for borders and navigation in conjunction to CSS styles.
- .Net controls: Pages and user controls will contain one or more controls that deal in the rendering of HTML.
- Master Pages: ProfilesWeb uses two Master Pages to define the layout of the site and each child page. ProfilesPageBase.master creates the layout for the header, footer and main body columns. ProfilesPage.master creates the layout of the details shown in each page (IE the menus,widgets and containers for each child page).

Updating Person Data After Go-Live

If you have frequently changing HR data, you can setup a recurring data load process. Each time you want to update Profiles, enter new data into the loading tables, and then execute the stored procedure usp_loadprofilesdata. For people or users who are already in Profiles, their data will be updated, but their PersonIDs and UserIDs will remain the same. If a person or user no longer appears in the loading tables, then the active flag in the Profiles database will be set to false, and their profiles will no longer be visible.

Proxies

Proxies are users who can permission to edit other people's profiles. There are two types of proxies, "designated" and "default". Designated proxies are people faculty assign to edit their profiles. Access is all or nothing, meaning if you assign someone as a designated proxy, he/she has full access to your profile.

Default proxies are "superusers" who have access to a large number of faculty by default. Default proxies can either be visible or hidden. When faculty go to their proxy page in Profiles, they can see any visible proxies who have access to their profiles. They cannot see the hidden default proxies. Default proxies can be assigned to multiple departments, institutions, or everyone. There are granular editing permissions for default proxies. For example, they can be allowed to edit everyone's publications, but only the photos for a particular department.

Default proxies must be entered directly into the database in the proxies_default table. The "proxy" field in this table is the proxy's Profiles UserID. The department and institution fields can either be empty strings, or a value can be given to either one to limit the population that can be edited by the proxy. The IsHidden field is "Y" if the person is a hidden proxies, and "N" otherwise. The "Edit" fields should be set to "1" if the proxy has permission to edit that content, and "0" otherwise. A single proxy can have multiple records in this table, where each record indicates scope and permissions for a different population of profiles.

Note that all proxies, both designated and default must have a record in the user table.

Federated Search (DIRECT)

Distributed Interoperable Research Experts Collaboration Tool (DIRECT) is a pilot project facilitated by the Research Networking Working Group of the NIH-supported Clinical & Translational Science Award (CTSA) Consortium to provide federated search across multiple institutions using different research networking tools. For more information about DIRECT, visit http://direct2experts.org.

Profiles now includes support for DIRECT, though it is disabled by default. In order to enable this feature, uncomment the following keys from the web.config file:

```
<add key="DirectServiceURL"
value="http://[YourDomain]/[YourProfileSubWebName]/DirectService.aspx"/>

<add key="DirectPopulationType" value="[Description of the population (e.g.,
faculty, staff, etc.) loaded into your instance of Profiles.]"/>

<add key="DirectQueryTimeout" value="15"/>
```

Replace the variables (in brackets) with the appropriate values. The database table FSSites lists the URLs of other DIRECT sites. This list changes frequently. Visit the direct2experts web site to learn how to join the mailing list to obtain the latest URLs.

Software Updates

If you are applying this update to an existing install, we recommend that you do the following:

1. Back up your existing web.config files first, and then re-apply them after you copy the new web code to your server to preserve previous settings.
2. If you have previously made customizations to the code, re-apply those changes to the new code rather than simply replacing the new files with your modified ones.
3. Stop and restart the IIS Application Pool to ensure the cache from the previous version of Profiles has been cleared.

We will be releasing bug fixes as they are identified for the next few months, but no major new features are planned for this current beta codebase. Profiles 1.0, which will replace the beta software, will be a complete rewrite. It will have a more modular architecture, enabling us to include many new features and to begin collaborative development with others who would like to participate in the open source project.

Release Notes

5/5/10   Fixed bug in cache_person update process that did not calculate cache_person.NumPublications correctly.

5/10/10 Fixed bug in usp_LoadProfilesData related to how the department table is populated.

6/20/10 Bug Fixes:

1. The co-author radial network browser does not load.
2. When a collaborator is deleted from the active network details page, it doesn't get removed from the left sidebar.
3. Clicking the link to view all people in one's department in the right sidebar (passive network) returns all people in the entire database.
4. Keyword categories page does not display all categories.
5. Inconsistent behavior across browsers with auto-complete on the main search form. [This feature has been removed for now.]
6. Back link on a person's profile to return to search results is broken.
7. Connection strength information on search results detail page is confusing. [The connection strength value has been removed for now.]
8. Faculty rank, not the affiliation title, is appearing as a person's title.
9. Faculty ranks are hard-coded in the website instead of being dynamically read from the database.
10. usp_loadprofilesdata - departments table isn't populated correctly during the load process.
11. Permissions are not set correctly for the default database application user account.
12. API does not handle secure mode and the isvisible person flag correctly.
13. Pubmed disambiguation process crashes if data is requested for a publication that has been deleted from Medline.
14. Search breaks if API cache tables are not purged. [Clear API cache tables process was added to the nightly database job.]
15. CSS fixes are needed for IE 6.

New Features:

1. Added global error logging to server event log and uniform custom error message display.
2. Added weightcategory tag to keyword XML (for use on keyword cloud page).
3. Added academicrank tag to person XML (a derived person level indicator of that person's highest faculty rank).
4. Created optional script ProfilesSetUserPermissions.sql to grant execute permissions to new database accounts.

Cosmetic:

1. usp_getusersupporthtml - added a wrapper div when edit mode = 1.
2. usp_parsepubmedxml - cleaned up code related to calculating a publication date.
3. fn_getuserdepartmentpeople - cleaned up code to better use cache tables.
4. Changed default database application user account to AppProfilesUser with password Pass1234.

6/25/10   Bug Fixes:

1. ProfilesNightlyJobs.sql fails.
2. View all people in a department returns the wrong list in certain cases.
3. Radial network browser is not working on certain browsers.
4. Google Map visualizations does not display markers for all people.
5. Page layout is incorrect on IE6 and IE7 due to CSS handling differences from other browsers.
6. Number of matching keywords on the Search Results "Why?" Connection page is incorrect for multiple keyword searches.

7/15/10   Bug Fixes:

1. Pagination is not working on the Search Results page when more than 15 items per page is selected.
2. Searches for names or keywords containing an apostrophe (O'Brien, Baker's Yeast) are not working.
3. IE 6 style conflict is causing Edit Profile page to indent far to the right.
4. Upload photo does not work on the Edit Profile page.
5. List view for Keywords Cloud tab redirects to Co-Authors List View tab.
6. usp_UpdateUnGeocodedAddress fails. [Changed to use addressstring, instead of derived address string.]
7. Exact Keyword search is not functioning as expected for multiple word phrases.
8. Clicking a "most viewed" item on the search page does not run an exact keyword search as expected.
9. The highest overall faculty/academic rank, rather than the rank of the primary affiliation should be displayed.

7/22/10   Bug Fixes:

1. usp_memebership_getpassword assumes that all users records have user.applicationname = 'Profiles'. This results in login failures for all users.

2. A user who clicks the "Most Viewed" keywords lists in the passive network margin, will trigger double quotes to wrap around the keywords, causing the counts to be incorrect in the search_history_kw table.

3. Trying to upload a photo while editing a profile fails.

8/9/10   Bug Fixes:

1. Dynamic Javascript in ProfilesEdit.aspx.cs is causing Javascript browser errors for IE 6.0 clients viewing profiles from IIS 6.0 servers.

2. Incorrect physical neighbors. [Added latitude and longitude to logic for determining physical neighbors (usp_cache_physical_nieghbors). Added indexes to cache_person for these columns to improve performance.]

3. Changed column/primary key for person_add table to internalusername.

4. Search returned an error if the same query had been done more than 24 hours earlier and the nightly jobs were not run. [Removed a date check in API (usp_getpersonxml_v2), which was used to retrieve cached query results.]

5. Removed mpid foreign key constraint on publications_add and publications_exclude as there is the possibility of null values.

6. API code is not reading IsSecure appSettings key in web.config file.

7. API does not allow search by InternalID.

10/27/10   Bug Fixes:

1. <dataConfiguration defaultDatabase="ProfilesDB"/> in line number 35 of the web.config file was incorrectly listed as <dataConfiguration defaultDatabase="[DB Name Here]"/> [This value needs to be set to ProfilesDB and match the ConnectionString key name of ProfilesDB.]

2. CoAuthorNet.aspx.cs: the inline test for if an address was null or the Latitude equal to zero causes a null exception error in the event address was equal to null. [This was corrected by splitting the test into two 'if'' statements, first test if pList.Person[0].Address != null then inside this condition, test if pList.Person[0].Address.Latitude == 0.]

3. When calling the Profiles API web service, the Response message is not returning the InternalID list. [The bug was in the project Connects.Profiles.Service.DataContracts, file name InternalId.cs, line number 56. The DataContract name was spelled with the incorrect case. The name should be InternalIDList with a capital "ID". The same applies to the DataMember name and Serialization.XMLElementAttribute in this file.]

4. Secure Socket (HTTPS) causes an error that prevents client side Ajax to execute and page content to display.

5. When you add a designated proxy to your profile and you search by institution, it never returns any records. [Changed usp_searchproxy to

remove the sub query select of {select name from institution_map where catalystname = @xInstitution} and changed it with {InstitutionFullName = @xInstitution}

6. usp_ProxySearch does not check for isActive status of the user. [Added AND isActive = 1 to the WHERE clause of usp_ProxySearch.

New:

1. vw_person preference defaults changed from Y to N for showphoto, showawards, shownarrative.

2. Added BSD license and Copyright header to all Web and Database code.

3. usp_GetPersonList_xml_V2 – added a join to the display_prefs table. Profiles will now use the display_prefs table directly instead of cache_person to enable user privacy settings to have an immediate effect.

4. usp_cache_sna_betweenness changed so that it is now calculating the exact betweenness score rather than an approximation.

5. usp_parseallpubmedxml proc changed to avoid foreign key conflicts when replacing pubmed data that already exists in publications_include.

6. usp_loadprofilesdata – changed default value of isactive column to 1 (active).

11/18/10   Bug Fixes:

1. Exception message: Could not find stored procedure 'dbo.aspnet_Applications_CreateApplication'. [This bug was introduced in the last release and has been fixed.]

2. In ProfilesSetUserPermissions.sql, the AppProfilesUser account does not have a sysadmin role.

3. In usp_LoadPMPublications_include the check to remove orphaned pubs didn't include publications_add. The result was that publications that were added were removed from publications_include after a disambiguation process.

12/20/10   Bug Fixes:

1. Inactive users are able to log into profiles. [A correction has been made to the usp_membership_getuserby name and usp_membership_getpassword stored procedures to check if the user is active.]

2. If the user does not have a profile they receive an error after trying to log on. [We now direct the person to the search screen if they try to log on and do not have a profile in the system.]

3. The proxy feature is not working correctly. I do not see my proxies after I add them to my profile. [A correction has been made for the display of Designated Proxies and Default Proxies so the correct parameters are passed into the usp_getproxies.]

4. The left sidebar was shifted to the right when viewing a profile with more than one affiliation. [Added a missing <Table> tag to the Separator Template in the repeater control in ucProfileBaseInfo.ascx]

39

5. The PubMed disambiguation process can fail if it is run after users have manually added publications to their profiles.

6. The PubMed disambiguation process occasionally failed unexpectedly. [The timeout on the Harvard-hosted disambiguation server was increased.]

7. Google Map for Similar People does not correctly display connections between people. [Issue was related to a bug in usp_getgmapusersimilar. There is a join from personid to userid in the main query. This was corrected to be personid to personid.]

3/17/11   Bug Fixes:

1. When user that does not have a profile selects "Edit my profile" from the active network menu, then logs into Profiles, a system error message is displayed. [A "ProfileExists" flag was added to the UserPreferences class. Then a test for ProfilesExists takes place in the OnLoad event of ProfilesEdit.aspx.cs file. If a user does not have a profile, the request is redirected to the Search.aspx page.]

2. When compiling the ProfileWeb project an error that License.txt is not part of the project occurs. [The Profiles project file meta data has been updated to reflect the correct the current BSD license.]

3. When running disambiguation jobs, packages sometime, intermittently timeout. [This is resolved by increasing the HTTPWebrequest object's timeout, in the script task, to 900,000 milliseconds or 15 minutes.]

4. In certain situations, searching for proxies would return inactive users. [This has been resolved by adding an isactive check in usp_getproxies and usp_proxysearch, for those cases.]

5. InternalLDAPUsername field length is too small. [Field length increased to 50 characters throughout the database.]

6. SQL Packages fail in SQL Server 2008. [Separate packages for SQL Server 2005 and 2008 are now available.]

7. Hide and show functions for Institution, Department, Division are not consistent between Mini Search and Search. Profiles now has 6 Flags to maintain the display of the 3 drop down controls on all search functions.

```
<add key="HideInstitutionSelectionForSearch" value="false"/>
<add key="HideInstitutionSelectionForMiniSearch" value="false"/>
<add key="HideDivisionSelectionForSearch" value="false"/>
<add key="HideDivisionSelectionForMiniSearch" value="false"/>
<add key="HideDepartmentSelectionForSearch" value="false"/>
<add key="HideDepartmentSelectionForMiniSearch" value="false"/>
```

New:

1. Search by Division has been enabled for both the mini search and the main search screen.

2. The federated query tool DIRECT has been added as an optional function to the Profiles Beta. See the above section on DIRECT for more information.

7/22/11   Bug Fixes:

1. Users would delete publications from their profiles, but they would be added back during the next automated disambiguation run.

2. The back-end process for adding publications fails if the database user is not give DDL permissions. [The code was changed so that standard read/write permissions are sufficient.]

3. The automated disambiguation process fails if users had previously added custom publications to their profiles.

4. Google maps sometimes displays the people at the wrong address.

5. Google maps sometimes does not display a pushpin.

6. When editing a previously-added custom publication, the publication type reverts to "Abstract" and other data fields disappear.

7. Manually adding publications using multiple authors fails in Firefox.

8. Letters 'kj' show up under "Most Viewed This Month".

9. On the search results page, when you mouse-over any researcher, the right pane display is switching the title and faculty rank values.

10. A user with profile does not always get the profile-owner menu.

New:

1. The source code has been moved to a Subversion repository hosted by Recombinant Data Corp.

2. A data validation stored procedure has been created to check the data in the loading tables.

3. The ability to hide a person's email and/or physical address has been added to the Edit Profile page. This feature is configurable via the web.config.

Known Issues

1. Certain browser security configurations (we haven't determined the exact settings) disable the JavaScript used by the drop down menus on the main search page to select More Options.

2. Selecting "Display Columns:" in the search results by clicking on the column name and not the checkbox fails to trigger a repost of the page.

3. Long item names in Passive Networks (right sidebar) are truncated.

4. The height of the More Options drop down menu on the search page does not dynamically adjust based on the number of items displayed.

5. When two browser windows or tabs are opened at the same time to Profiles, it can create inconsistencies with internal session variables. This can be reproduced by running a search in window #1, then running a different search in window #2, then clicking the "Why?" link in the search results of window #1.

6. When editing a profile in Mac FireFox or Mac Chrome, users cannot add new awards.

7. Network View sometimes fails to load in Mac Opera.

8. Uploading of profile photos on ProfileEdit.aspx causes a javascript/AJAX permission denied error if the client browser is IE 6.0 and the web server is IIS 6.0

9. Rarely, certain PubMed articles cause the disambiguation process to hang or fail.

Acknowledgements

Harvard Development Team

The software implementation is led by the Harvard Medical School Information Technology Department. The current and past members of the development team include Nick Benik, Niraj Desai, Paul Gomez, John Halamka, Ken Huling, Shashank Jain, Melissa Kenny, Kevin Laitinen, Kellie Lucy, Krishna Nellutla, James Norman, Rob Piscitello, George Rakauskas, Jeff Rosen, Michele Sinunu, Franco Valentino, Marlon Violette, Griffin Weber, and Steve Wimberg.

UCSF Development Team

The UCSF Profiles team includes Mini Kahlon, Eric Meeks, Kristine Moss, Rachael Sak, and Leslie Yuan. UCSF has developed innovative promotional strategies for research networking, assisted with quality assurance, and are adding OpenSocial support to Profiles RNS. Mini Kahlon is co-chair with Griffin Weber of the National CTSA Research Networking Group, which is leading the efforts to create a national pilot to demonstrate interoperability among different research networking platforms.

Recombinant Data Corp.

The Profiles RNS team at Recombinant Data Corp. includes Kimber Barton, Peter Emerson, Dan Housman, Mike Klumpenaar, Mark Mischke, Matvey Palchuk, and Nancy Pickard. Recombinant provides commercial support for Profiles RNS, hosts publication disambiguation services, develops administrative tools for Profiles RNS, and writes documentation (including portions of this install guide) and marketing materials.

Profiles RNS Users Group

We thank the member institutions of the Profiles RNS Users Group for their willingness to be early adopters of the software and their continued feedback. For a list of member sites, please visit the Community page on http://profiles.catalyst.harvard.edu.

More Information

For more information about Profiles RNS, please visit

http://profiles.catalyst.harvard.edu

The Harvard development team can be reached at profiles@hms.harvard.edu. We will try to reply promptly, though we cannot guarantee that we will be able to answer all questions.

Commercial support options are available through Recombinant Data Corp. Harvard has no financial relationship with Recombinant, but we recommend them as an Authorized Service Provider for Profiles RNS. For more information, contact Recombinant at results@recomdata.com or call (617) 243-3700.