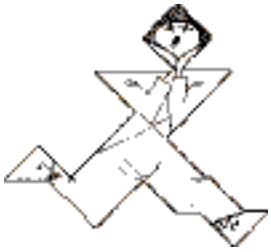


Hay River Software LLC

ForXs2gApps™ User Management agent for Google Apps for Education®

Hay River Software LLC N14397 380th St, Ridgeland WI 54763 **T** 715 455-1652 george@hayriv.com http://hayriv.com

Legal Notice

Hay River Software LLC N14397 380th St, Ridgeland WI 54763 **T** 715 455-1652 **george@hayriv.com** **http://hayriv.com**

ForXs2gApps

User Management agent for Google Apps for Education.

Introduction

Important Note: This documentation is for the beta version of ForrXs2gApps. It has been adapted from the documentation for the original ForXs agent for Macintosh OS X server. There may be errors in this document, missing information or statements that don't apply to this beta agent. If any discrepancies are found please contact george@hayriv.com.

All of the information about using this agent in a SIF environment has been eliminated from this document to make the documentation as concise as possible. Please refer to the ForXs2gAppsSIF.pdf document for instructions on setting up the agent as a SIF agent.

About ForXs2gApps agent

ForXs2gApps is tool to automate user creation in Google Apps for Education or Google Apps for Enterprise.

As a SQL based management agent

ForXs2gApps queries a SQL database periodically to get at least first name, last name and a unique number, usually a student number or staff ID . It uses this information to create and manage user accounts in Google Apps for Education. Configuration settings allow the agent to accept usernames and passwords in the queries or it can create unique usernames and passwords as needed. The frequency of the queries can be configured anywhere from every second to daily.

Planning and Configuration

Requires a computer with Java 1.5 or newer installed.

Installing ForXs2gApps

ForXs2gApps.jar

ForXs2gApps can be downloaded from http://www.hayriv.com/ForXs2gApps/downloads.html. The files are in .zip format. They should expand automatically if you are running Macintosh System 10.4 or newer. ForXs2gApps is a Java runnable jar file along with a config folder and work folder. These files need to stay in the same folder.

Activating, Running, and Deploying ForXs2gApps

Minimal required setup.

The following steps need to be completed before users can be created using ForXs2gApps.

License Key

An Application License Key is required to use the ForXs2gApps application. The file name is ForXs.key. The key is an encrypted file that unlocks the application and will work only for your computer. This key file needs to be located in the same folder as the ForXs2gApps application. If not found there the application will ask for its location and move it for you. Check out the ForXs2gApps License page for more information and instructions for obtaining a key. http://hayriv.com/ForXs2gApps/license.html

Setting Preferences for ForXs2gApps

All preferences are set by editing a XML file with the default name of config.xml. Instructions for editing the file follow on this page or in later sections of this document.

Important Note

Test this software with

```
<property name="gmailAllowCreation" value="false"/>.
```

This will allow the populating of the internal database before accounts are actually created in Google Apps for Education. For security reasons a username once deleted in Google Apps can not be recreated for 5 days.

Do not test this software on a server that is being used for active production purposes. This agent will reset the password for an existing user if that username already exists which would make the user unable to log in to their account.

Prerequisites

1. A SQL table with staff or student information.
2. The ForXs2gApps software, available for download at http://hayriv.com/ForXs2gApps/downloads.html
3. A license key for the ForXs2gApps software, available at http://hayriv.com/ForXs2gApps/license.html
4. Java version 5 installed (1.5) On a Apple OS X 10.4 or higher this should be installed by default if the normal Apple Software Updates are up to date.

Partially configured files

A basic configuration file is provided for these student information systems in the config/sample configurations/SQL folder of ForXs2gApps;

Replace the config.xml file in the ForXs2gApps/config folder with the file associated with your database system.

Google Apps for Education connection configuration.

Google config.xml file settings here;

```
<property name="gmailAdminEmail" value="admin@myschooldomain.k12.mn.us"/>

<property name="gmailAdminPassword" value="theAminPassword"/>
```

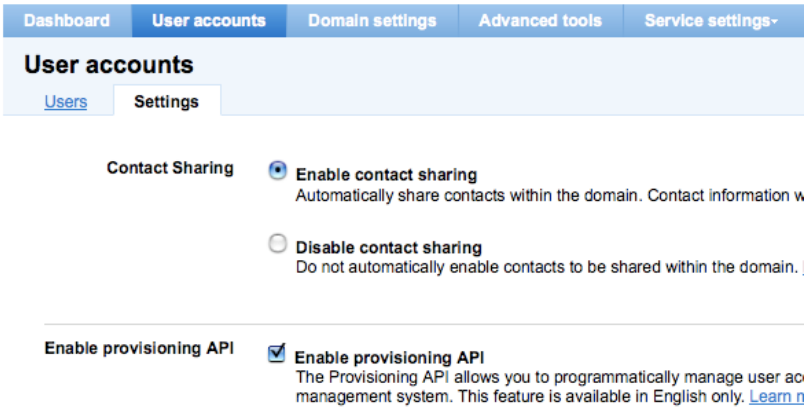
```
<property name="gmailDomain" value="myschooldomain.k12.mn.us"/>

<property name="gmailSetChangePasswordAtNextLogin" value="true"/>

<property name="gmailAllowCreation" value="true"/>
```

The gmailAdminEmail address is the full email address of an administrator including the @domain..
gmailAdminPassword is the admin's password.
gmailDomain is the portion of the email address after the @ for your domain.
gmailSetChangePasswordAtNextLogin when set to true will force users to change their password on the next log in. This is a best practice.
gmailAllowCreation will need to be set to true to allow users to be created but this should be set to false when first testing the agent. This will allow you to verify that usernames and passwords are being generated in a manner that you want before creating many users that will need to be deleted.

The provisioning API for your Google Apps for Education also needs to be turned on before this agent can communicate and create users. To turn this on log into your Google domain's management web page. Select the "User accounts" tab and then the "Settings" sub tab. Click the "Enable provisioning API" checkbox and save your changes. The picture below shows what this page looks like.



Internal database configuration.

```
    <property name="hibernate.connection.driver_class" value="org.hsqldb.jdbcDriver"/>

    <property name="hibernate.connection.url" value="jdbc:hsqldb:work/ForXsdata"/>

    <property name="hibernate.connection.username" value="sa"/>

    <property name="hibernate.connection.password" value=""/>

    <property name="hibernate.connection.dialect" value="org.hibernate.dialect.HSQLDialect"/>
```

If you will be writing SQL queries on your own database of users then you will need to change the ForXs database and configure it to use the same type database with a user that has rights to read your database. An example of connecting to a MS-SQL database is listed below. The ForXs data and your user data can reside in different databases as long as those databases can be accessed with a join in your SQL query.

```
    <property name="hibernate.connection.driver_class" value="net.sourceforge.jtds.jdbc.Driver"/>
```

```
    <property name="hibernate.connection.url" value="jdbc:jtds:sqlserver://yourDatabaseIPorURL:
1433/databasename"/>

    <property name="hibernate.connection.username" value="yourUsername"/>

    <property name="hibernate.connection.password" value="yourPassword"/>

    <property name="hibernate.connection.dialect" value="org.hibernate.dialect.SQLServerDialect"/
>
```

SQL queries to get new, modified or unenrolled students.

Three SQL queries need to be written to connect your SQL data source to the ForXs SQL tables. These queries are stored in the config file as sqlSourceAddQuery, sqlSourceChangeQuery and sqlSourceDeleteQuery. They need to return information in a specific format which will be explained below. An empty string or '' is returned for those items not available or supplied in the query. It is normally best to write and test these queries in your normal query tool and paste the result into the configuration file. When testing limit the number of records returned from the queries using the TOP or LIMIT SQL commands. Below are examples of accessing a MS-SQL database.

```
    <property name="sqlSourceAddQuery" value="SELECT   TOP 5  sa.studentnumber AS otherID,
sa.firstname ,  sa.lastname ,  sa.username,  sa.password,  sa.schoolnumber AS schoolInfoRefID,
sa.grade,  sa.gradyear,  '' AS RefID,  sa.birthDate,  sa.streetNumber,  sa.phone AS phoneNumber,
sa.middleName,  sa.personID   FROM campusrpts2005.dbo.StudentAccounts sa  left outer join
userinfo ui on ui.otherid = sa.studentnumber WHERE sa.disable !='1'  AND ui.otherid is null "/>

    <property name="sqlSourceChangeQuery" value="SELECT TOP 5   sa.studentnumber AS otherID,
sa.firstname ,  sa.lastname ,  sa.username,  sa.password,  sa.schoolnumber AS schoolInfoRefID,
sa.grade,  sa.gradyear,  '' AS RefID,  sa.birthDate,  sa.streetNumber,  sa.phone AS phoneNumber,
sa.middleName,  sa.personID   FROM campusrpts2005.dbo.StudentAccounts sa  join  userinfo ui on
ui.otherid = sa.studentnumber WHERE sa.disable !='1'  AND ( ui.lastname != sa.lastname OR
ui.firstname !=  sa.firstname  OR ui.grade !=  sa.grade OR ui.schoolInfoRefID !=  sa.schoolnumber
OR ui.password !=  sa.password OR ui.gradyear !=  sa.gradyear ) "/>

    <property name="sqlSourceDeleteQuery" value="SELECT TOP 5 sa.studentnumber AS otherID,
sa.PersonId ,  '' AS RefID FROM campusrpts2005.dbo.StudentAccounts sa  join userinfo ui on
ui.otherid = sa.studentnumber  WHERE sa.disable ='1' OR sa.endDate != null OR sa.endDate &lt;
getDate() "/>
```

The sqlSourceAddQuery query.

The result of this query should contain the following columns;

otherID, firstName, lastName, middleName, username, password, schoolInfoRefID, grade, gradyear, RefID, birthDate, streetNumber, phoneNumber, personID

Only the otherID, firstName and lastName are required to have values. The rest can be returned using the '' AS fieldName syntax.

So that the agent doesn't get the same users time and time again the query does a LEFT OUTER JOIN with the userInfo table and the WHERE clause limits the results to otherIDs that are null. Once a user is created there will be a corresponding otherID in the internal ForXs database and this record will not be considered new. Here is the above example with comments.

```
SELECT   TOP 5  -- when testing just ask for a few records
sa.studentnumber AS otherID,  -- the database has a studentnumber but it needs to be returned as otherID
sa.firstname ,  sa.lastname ,  -- these are the same in both databases so no AS is needed.
sa.username,  --this database is supplying username otherwise it would be '' AS username
sa.password,
sa.schoolnumber AS schoolInfoRefID,  --the schoolnumber will be used to JOIN with the schoolInfo table
sa.grade,
sa.gradyear,
'' AS RefID,
sa.birthDate,  --this should be text and formatted at YYYYMMDD
sa.streetNumber,  --this can be either the address number or the entire street with address
sa.phone AS phoneNumber,
sa.middleName,
sa.personID --this is only used with Infinite Campus SIS so would normally be '' AS personID
FROM campusrpts2005.dbo.StudentAccounts sa  --the linked database where or info is coming from
LEFT OUTER JOIN  userinfo ui on ui.otherid = sa.studentnumber  -- Join to the userInfo table
WHERE sa.disable !='1'  --the disable column in this table is for users we don't want to create.
--Your DB will be different
AND ui.otherid is null -- this is done so we don't get users that are already created.
```

This query will be pasted into the config.xml file and there are a few things that won't work in XML that might seem OK in your query tool.

1. Don't a in comments as the query might get rewritten as a single line and then the comment wold keep the rest of the query from working.
2. Use != instead of <> for not equal in your queries
3. Use < in place of <
4. Don't use a double quote " in the query as this indicates the end of the value=" in the XML. Using " might work as an alternative. (not tested)

The sqlSourceChangeQuery query.

The result of this query should contain the following columns;

otherID, firstName, lastName, middleName, username, password, schoolInfoRefID, grade, gradyear, RefID, birthDate, streetNumber, phoneNumber, personID

Only the otherID, firstName and lastName are required to have values. The rest can be returned using the '' AS fieldName syntax. The SELECT and FROM portions of this query will match that of the sqlSourceAddQuery except the a normal JOIN is used instead of a LEFT OUTER JOIN. The WHERE portion will look for changes between the two tables.

```
SELECT TOP 5
sa.studentnumber AS otherID,
sa.firstname ,
sa.lastname ,
sa.username,
sa.password,
sa.schoolnumber AS schoolInfoRefID,
sa.grade,
sa.gradyear,
'' AS RefID,
sa.birthDate,
```

```
sa.streetNumber,
sa.phone AS phoneNumber,
sa.middleName,
sa.personID
FROM campusrpts2005.dbo.StudentAccounts sa
JOIN  userinfo ui on ui.otherid = sa.studentnumber
WHERE sa.disable !='1'  AND
( ui.lastname != sa.lastname
OR ui.firstname !=  sa.firstname
OR ui.grade !=  sa.grade
OR ui.schoolInfoRefID !=  sa.schoolnumber
OR ui.password !=  sa.password )
```

In the case above the record will be considered changed if any one of the fields changes, lastname, firstname, grade, schoolnumber, or password.

The sqlSourceDeleteQuery query.

The result of this query should contain the following columns;

otherID, personID, RefID

```
SELECT TOP 5
sa.studentnumber AS otherID,
sa.PersonId ,
'' AS RefID
FROM campusrpts2005.dbo.StudentAccounts sa
JOIN userinfo ui on ui.otherid = sa.studentnumber
WHERE sa.disable ='1'
OR (sa.endDate != null
AND sa.endDate &amp;lt; getDate() )
```

In the example SQL query above an account will be suspended if the database disable field is set to 1 or if the student is no longer enrolled as indicated by by the endDate field no longer being empty and the date being some date before to-days date.

Save config.xml

Save the configuration file as config.xml in the config folder is the same folder as ForXs2gApps.jar file.

Further configuration after initial testing.

There are many more options that can be changed to make the agent create users in various ways. The default settings in this file should work for initial testing purposes. After the initial testing is successful read through all the other sections in this document before creating all your user accounts. These sections go into greater detail for options available.

Testing ForXs2gApps

Prerequisites

ForXs2gApps software has been downloaded and configured per the instruction on the previous page.

License file.

Start up the agent. After starting up it will ask for the license file. Browse the hard drive for the file and select it.
Quit the agent. Restart the agent.

Check Connections

Check the Status window to see if connections to the Google server, and database worked. Messages similar to the following should appear.

Internal database

Successful connection to the internal database.
"Connecting to internal database"
followed by no error messages.

Google Apps connection

A successful connection to the Google server will produce the following message in the ForXs.log file and in the Status window.
"Successfully authenticated to Google Apps feed services." An error message will appear in it's place if the settings are not correct or if the Google provisioning API has not been enabled.

Failed connections?

If any of the three connections were not successful take note of the error messages, quit the agent and make adjustments to the config.xml file.
If the agent didn't show any windows or you need more information on the errors, open and check the end of the console.log, ForXs.log and the ForXsHibernate.log in the ForXs2gApps/work/ folder for more details.

Check for users in the internal database

The sqlQueryTimer will start up and execute the first query in 30 seconds after the timer has started. A successful query will report the number of users created or will report that no records were found. Checking the internal database should show new records being added. At first there should be no records to change or delete. These queries need to be checked after users have been created by changing the source database. SQL errors will get reported in the Status window, the ForXs.log or the ForXsHibernate.log file. If changes need to be made to the SQL queries, quit the agent, make changes to the config.xml file and then start up the agent again.

Allow the agent to create accounts in your Google Apps for Education site.

Once the records being created match what you expected you can stop the agent and change the gmailAllowCreation property to true.

```
<property name="gmailAllowCreation" value="true"/>.
```

Restart the agent. As new users come into the agent there should be an indication that the users are being created in your Google domain. It takes about 3 to 5 seconds to create each account. Log into you Google domain administration and check for new uses.

Password Lookup

Google Apps setup

Google Apps for Education domains are set up with a password scheme that is very secure. Once a password is set or reset there is no way to get that password back in a human readable form. If your preferences in ForXs2gApps are set up to "Create Random passwords" it would be very hard to ever figure out what the passwords are.

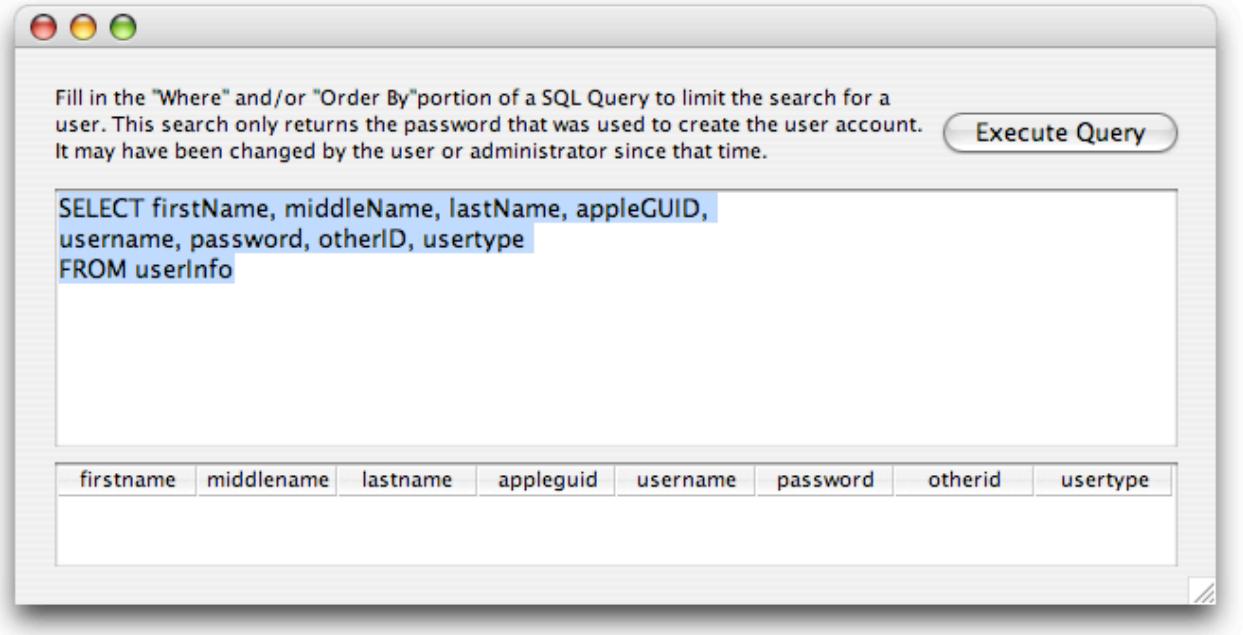
ForXs2gApps agent data tracking

ForXs2gApps maintains a internal database of user information. This information is collected as results from SQL come in. The information is checked and when all elements are recorded a user account is created. The passwords in this database are not encrypted.

Accessing the user information

You can use the simple query tool built into the agent or use the query tool you normally use to access your student information. The ForXs2gApps agent creates two table to store information, userInfo and schoolInfo.

When ForXs2gApps is running select the menu item "Search PW DB" available under the Window menu.



SQL Queries

The window opens with the following query.

```
SELECT firstName, middleName, lastName, grade, appleGUID,
username, password, otherID, usertype
FROM userInfo
```

This will return a limited amount of information about all users in the database. The SQL query can be altered to find a small subset of users or to display all information collected.

Example query to a password for a particular person.

```
SELECT password From userinfo WHERE firstName = 'George' and lastName = 'Adams'
```

Example query to find all fields and all records

```
SELECT * FROM userinfo
```

Example query to change the value of a single field in a record

```
UPDATE userinfo SET username='georgea003' WHERE otherID = '123456'
```

Example query to delete a record

```
DELETE FROM userinfo WHERE otherID = '123456'
```

Depending on the setup of your preferences deleting a user in this database and not in the server could possibly allow two user records to be created for a single user. Users deleted in this database should also be deleted from the server using Google Apps for Education administration web pages.

Example query to delete all records

```
DELETE FROM userinfo
```

If all users are deleted from the Google Apps at the beginning of the school year you may also want to delete all the records in this database so all information is requested fresh from the zone integration server. Depending on the username creation scheme deleting all the users in the internal database could create usernames with different digits at the end, sfranks02 instead of sfranks01.

Exporting Information

Exporting this information is not built into the agent but the data returned from a SQL query can be selected and copied using the shortcut keys Open Apple/Command and “C”. The result can then be pasted into a text document. As an alternative use the query tool you would normally use to access your database of student information.

Configuration of User Passwords in the ForXs2gApps agent.

Simple static passwords

Choosing static passwords will create the same password for every user. There are few if any situations where this is a good idea. The dynamic static passwords below is generally a better solution but much of the configuration is the same. In the following set of properties PwUseStatic should be set to true and the other two to false.

```
<property name="PwUseStatic" value="true"/>
<property name="PwCreateRandom" value="false"/>
<property name="PwUseSIF" value="false"/>
```

Change the value of the "noStaticPasswordSet" to be your static password;

```
<property name="PwStatic" value="noStaticPasswordSet"/>
```

When using simple static passwords the model user should have the password policy set to require a password change on next log in otherwise any student could log in as any other user.

Complex static passwords

Complex static passwords are a step up from the plain static passwords above. The property settings are the same as those above but the PwStatic property value is replaced with a combination of words that will be replaced with information from the students data.
The following "swap" words will be replaced in the "static" password.
GradYear, GradeLevel, OtherId, First1, Middle1, Last1, BirthDate, StreetNumber, PhoneNumber.

GradYear will be replaced with the 4 digit year the student is expected to graduate.
GradeLevel will be replaced with the student's current grade.
First1 will be replaced with the first letter of the first name.
Middle1 will be replaced with the first letter of the middle name.
Last1 will be replaced with the first letter of the last name.
OtherId will be replaced with the student number used as the uid for the user account.
BirthDate will be replaced by the students or staff members birthdate in YYYYMMDD format.
StreetNumber will be replaced by the number portion of the address.
PhoneNumber will be replaced by the phone number as formatted in the SIF XML message.

Note that these swap words are case sensitive and will not be replaced if they don't match. Last1 will be replaced but last1 will not be replaced.
GradYear and GradeLevel are replaced with "staff" for staff members and "employee" for employees.
OtherId for staff members will be their staff id numbers proceeded by the number 9 unless the default settings are changed.

If no data was received for a student for one of the swap words the words will be replaced with an empty space. For this reason it would be wise to have some characters in the "static" password other than just the key words. "*BirthDate*" for a static password would create a password of "**" if no BirthDate was recorded for a student but would look like "*19921205* if they did have a BirthDate that was passed by SIF.
Note that Google requires passwords to be at least 6 characters in length.

Random passwords (default setting)

In the following set of properties PwCreateRandom should be set to true and the other two to false.

```
<property name="PwUseStatic" value="false"/>
<property name="PwCreateRandom" value="true"/>
<property name="PwUseSIF" value="false"/>
```

The following properties value will be used as a source of the characters used to create passwords.

```
<property name="PwOkCharacters" value="abcdefghjkmnpqrstuvwxyz23456789"/>
```

The default value above was chosen because the letters are not easily confused with other. The letters i, I, l, L and the number 1 have been removed from the list as have o, O and zero. To make the passwords more complicated the following letters could be added. Upper case letters, !@#$%^&*()_+<>?~{}[]\. Letters could also be added in multiple times to make them more likely to appear as they do in normal words, eeeaaassttt.

The PwLength property determines how many "random" characters will be in the password. Google Apps requires a minimum of 6 characters.

```
<property name="PwLength" value="7"/>
```

In order to figure out what the passwords are someone will need to query the internal database. Two properties will allow this to happen and should be set as follows;

```
<property name="PwEraseAfterUserCreated" value="false"/>
<property name="dbHidePasswords" value="false"/>
```

Once the passwords are known dbHidePasswords can be set back to true.

Password supplied in SQL query

In the following set of properties PwUseSIF should be set to true and the other two to false.

```
<property name="PwUseStatic" value="false"/>
<property name="PwCreateRandom" value="false"/>
<property name="PwUseSIF" value="true"/>
```

Other Password settings

PwEraseAfterUserCreated will delete the password from the internal ForXs2gApps database after the password has been set in Google Apps. This option should **not** be used when creating random passwords as there would be no way to know what the passwords for users are. This option should be set to true when passwords are provided by another SIF agent.

```
<property name="PwEraseAfterUserCreated" value="false"/>
```

dbHidePasswords will display passwords as **** when set to true. This would make the passwords slightly more secure as the casual user would not be able to read a password.
<property name="dbHidePasswords" value="false"/>

Configuration of Usernames in the ForXs2gApps agent.

ForXs2gApps created usernames

The ForXs2gApps agent can create unique usernames, or short names based on the first, middle and last names of your students and staff. This would be the option chosen if there was no username supplied in your SQL query. Even if usernames are available you might want to leave them out of the SQL query so new different usernames could be created. Having a different combination of username and password would allow you to manage other password systems by being able to automate emailing lost passwords to an email account. If all usernames and passwords are unified including email there is no easy way to get forgotten usernames and passwords out to users.

Created username configuration

To have ForXs2gApps create usernames UnUseSif should be set to false

```
<property name="UnUseSif" value="false"/>
```

UsernameFormat sets the alpha portion of the username. Before creating usernames the First, Last and Middle names provided are made all lower case and all characters other than a-z are removed. Fs,Ms and Ls in the format string will be replaced with letters from the First, Middle or Last name. Any F,M or Ls in excess of the length of the users name are ignored. Letters other than f, m and l will be left in place. Don't use spaces.
Examples
Given the name Laura Jane Smith-Jones
format fffffffffffffl the alpha portion of the username would be "lauras".
format lllllllll_f the alpha portion of the username would be "smithjon"_j
Given the name Kay Ray Fay
format ffffffff-mmmmmm-llllll_student the alpha portion of the username would be "kay-ray-fay_student" .

```
<property name="UsernameFormat" value="ffffffffffffl"/>
```

Things to consider when deciding on a username format;

A short username is easier and faster to type but will be less informative when used as an email address. It can also cause more of the alpha portions of the username to match so more trailing digits may need to be added to make it unique.
It might be easier to locate if the last name proceeds the first name in the UsernameFormat.

Digits added to usernames

Usernames are made unique by adding digits to the end of alpha portion of the username. How the digits are added is determined by three properties UnDigits, UnPadWithZeros, UnRandom.

UnDigits sets the range of digits that will be added. 1 uses numbers 1 through 9 and should only be used in schools with very small populations. 2 uses numbers 1 through 99 and 3 uses numbers 1 through 999. Unless very short usernames are used and you have a very large population number above 3 would not be needed.

```
<property name="UnDigits" value="3"/>
```

UnPadWithZeros can be set to true or false. When set to true leading zeros are added to all numbers to make them the same length as UnDigits above. Set to true a username would look like "myang002" while if set to false it would be "myang2".

```
<property name="UnPadWithZeros" value="true"/>
```

UnRandom can be set to true or false. When set to true each new username is given a trailing number picked randomly from the range of possible numbers as determined by UnDigits above. If it is set to false numbering starts at one and increments up. The false setting makes for a large number 01 or 001s. It might be a problem as many students could be easily singled out by figuring out the username scheme and adding a 001 to the end. When creating a random number the agent checks both the internal database and the existing user accounts to make sure it doesn't create accounts with duplicated usernames.

```
<property name="UnRandom" value="true"/>
```

Username supplied by SQL Query

To have ForXs2gApps use usernames supplied to it by SIF UnUseSif should be set to true

```
<property name="UnUseSif" value="true"/>
```

The property convertSifUsernamesToLowercase when set to true will convert usernames with uppercase letters to lowercase. Email usernames are generally lowercase. When logging in usernames are not case sensitive but passwords are. Setting convertSifUsernamesToLowercase to false will leave usernames as they are received in the SQL query.

```
<property name="convertSifUsernamesToLowercase" value="true"/>
```

When using SQL for usernames the following properties are ignored; UsernameFormat, UnDigits, UnPadWithZeros, UnRandom

Configuration of Traffic Management in the ForXs2gApps agent.

Properties explained

```
<property name="shellTimerFrequency" value="5"/>
<property name="sqlSourceQueryFrequency" value="300"/>
```

Overview

Some school districts have concerns about the amount of network traffic created by SQL queries, the added load on the student information system. The properties above can be set to control the actions of the agent to ensure the agent has a minimal impact. There are two threads or separate sets of actions. The timing of each thread can be controlled.

SQL queries to check for new, modified or unenrolled users

```
<property name="sqlSourceQueryFrequency" value="300"/>
```

sqlSourceQueryFrequency determines how often in seconds three queries are executed to get information about new, modified or unenrolled students. These can be resource intensive queries and should only be done as often a necessary.

User Creation based on changes to the internal database

```
<property name="shellTimerFrequency" value="5"/>
```

The ForXs2gApps creates the user as soon as all information is available. It sets the passwordStatus field in the database to "created". The shellTimer thread, working at its own pace, queries the internal database looking for a passwordstatus of "ready". If the internal database is updated by setting the passwordstatus to "ready" this thread will create the users. The shellTimerFrequency property determines the delay in seconds between queries to the database to see if any new passwords are ready to be created.

```
<property name="shellTimerFrequency" value="5"/>
```

The value is in seconds and can be adjusted as needed.

Configuration of Graphical User Interface in the ForXs2gApps agent.

Properties

```
<property name="screenLogLevel" value="all"/>
<property name="adk.log.level" value="-1"/>
<property name="dbHidePasswords" value="false"/>
<property name="ShowProperties" value="false"/>
<property name="runHeadless" value="false"/>
```

screenLogLevel

This property controls how much information is displayed in the Status Window. Valid options are none, info, warnings, debug, all. By default it is set to debug but should be set to info or none once testing and synchronization are complete.
<property name="screenLogLevel" value="all"/>

adk.log.level

This property determines how much data is written into the log file. Some of the SIF specific and XML messaging information is only written to this log and not to the Status Window. Valid options are the numbers listed below. The numbers can be added together to combine options.
ADK.DBG_TRANSPORT is 4
ADK.DBG_MESSAGING is 8
ADK.DBG_MESSAGING_PULL is 64
ADK.DBG_MESSAGING_DETAILED is 128
ADK.DBG_MESSAGE_CONTENT is 256
ADK.DBG_PROVISIONING is 512
ADK.DBG_LIFECYCLE is 268435456
ADK.DBG_PROPERTIES is 1073741824
The following are combinations of the above
ADK.DBG_ALL is -1
ADK.DBG_NONE is 0
ADK.DBG_MINIMAL is 536871424
ADK.DBG_MODERATE is 805306888
ADK.DBG_MODERATE_WITH_PULL is 805306952
ADK.DBG_DETAILED is 805307084
ADK.DBG_VERY_DETAILED is 1879048956

```
<property name="adk.log.level" value="-1"/>
```

dbHidePasswords

This property will display passwords as **** when set to true. This would make the passwords slightly more secure as the casual user would not be able to read a password. The default value is true. Valid options are true or false.

```
<property name="dbHidePasswords" value="false"/>
```

ShowProperties

This property will display properties as they are read in by the agent. This can help debug problems where a property has been added to a file twice and only one value is being read in. The default value is false.

```
<property name="ShowProperties" value="false"/>
```

runHeadless

This property will cause the agent to start up and not display any windows or menus. The default value is false. It can be set to true when all configuration is complete and the agent is set up to run as a daemon. If the agent encounters a problem with license file or config file when started it will temporarily set this property to true and display the windows and menus.

```
<property name="runHeadless" value="false"/>
```

calcGradYearFromGrade

For those student systems that don't supply the GradYear element this and the next property can be used to calculate a graduation year. The default is false which will allow the StudentPersonal GradYear element fill the GradYear.

```
<property name="calcGradYearFromGrade" value="false"/>
```

currentGradYear

This property is used when calcGradYearFromGrade is set to true. The default is 2007 but should increase by one each school year. This should be 2007 for the 06-07 school year. The calculation used is
currentGradYear + 12 - GradeLevel = GradYear

```
<property name="currentGradYear" value="2007"/>
```

Working with the internal database of users

How it works

In a SIF environment SIF sends out chunks of information in SIF objects. No single object has all the information needed to create a user so ForXs2gApps holds what it does get in an internal database until it has all the pieces then it creates the user and shortly after that creates the password. When using SQL queries to get information all the needed information would normally be received at one time but the agent still records this information to aid in creating unique usernames.

Troubleshooting

The initial set up of ForXs2gApps and its connection to a set of SQL queries may produce unwanted results. The database can be used to see if the information you expected to be receiving has actually arrived and been recorded by ForXs2gApps. Use the sample SQL queries below to display data from the database.

Exporting from ForXs2gApps

There is no built in feature to export data from the database but you can run a SQL query, select all the data, copy it and then paste it into a text document. It produces a tab delimited file which can be read by other programs. Note that the passwords in the database are those that ForXs2gApps used to create the initial password for the user and if the user changes their password it will not be reflected in this database,

SQL Queries

This is the default SQL query that shows up when first opening the PW DB window.

```
SELECT firstName, middleName, lastName, grade, appleGUID,
username, password, otherID, usertype
FROM userInfo
```

Display every user and all columns in the database

```
SELECT * FROM userinfo
```

Display all usernames and passwords for students only

```
SELECT username, password FROM userinfo WHERE usertype like 'student%'
```

Delete all the records without deleting the database.

```
DELETE FROM userinfo
```

If you are doing this to start a new year fresh you might want to do the same in Workgroup Manager by deleting all the user accounts other than the admin accounts and model user accounts.

Delete all the passwords in the database.

```
UPDATE userinfo SET password = ''
```

Reset a password for a particular user and force the agent to change the password in Google Apps.

```
UPDATE userinfo SET password = 'apple' , passwordstatus='ready' WHERE username='fredh001'
```

Check if GradYear is coming from a SQL query or being calculated based on Grade

```
SELECT gradYear, grade, firstName, lastName
FROM userinfo
ORDER BY gradYear
```

Find a particular student by student number

(replace the 123456 with the number you are looking for)

```
SELECT firstName, middleName, lastName, grade, appleGUID, username, password, otherID, usertype
FROM userInfo
WHERE otherID ='123456'
```

Find all staff users sorted by last name

```
SELECT firstName, middleName, lastName, grade, appleGUID, username, password, otherID, usertype
FROM userInfo WHERE usertype LIKE 'staff%'
ORDER BY lastName, firstName
```

Count the number of user records in the database.

```
SELECT count(*) FROM userinfo
```

Count the number of student user records in the database.

```
SELECT count(*) FROM userinfo WHERE usertype LIKE 'student%'
```

Count the number of staff user records in the database.

```
SELECT count(*) FROM userinfo WHERE usertype LIKE 'staff%'
```

School Info Table

This table will be filled or updated any time the zone is queried for SchoolInfo

To display all school records

```
SELECT * FROM schoolinfo
```

To delete all school records

```
DELETE FROM schoolinfo
```

To delete some school records

```
DELETE FROM schoolinfo WHERE schoolname LIKE '%elem%'
```

This would find and delete all the elementary schools that actually had Elementary in their name
It might also delete a High School if its name was Celement High so be careful.

Password options involving keywords swapped for user information.

Swap words

There are a number of "swap words" that can be added to complex "static passwords". These specific words will be replaced in a new user's password when the user is created. There are two categories of swap words. The first is user information and the other is school information.

Places where words are swapped

In the password, when using a static password for all users.

User Information Swap words

The following user swap words replace the word with information specific to that user.

- **GradYear** is replaced with the graduation year supplied in the StudentPersonal SIF object.
- **GradeLevel** is replaced with the current grade level supplied in the StudentSchoolEnrollment SIF object.
- **OtherId** is replaced with the student number in the StudentPersonal object or by the staff number for Staff members
- **First1** is replaced with the first letter of the first name of the student, staff or employees name.
- **Middle1** is replaced with the first letter of the middle name of the student, staff or employees name.
- **Last1** is replaced with the first letter of the last name of the student, staff or employees name.
- **BirthDate** will be replaced by the students or staff members birthdate in YYYYMMDD format.
- **StreetNumber** is replaced with the number portion of the user's street address. *
- **PhoneNumber** is replaced with the phone number of the user as supplied by SIF. It is replaced in the same format as supplied.

Note that these words are case sensitive and will not be replaced if they don't match. Last1 will be replaced but last1 will not be replaced.
GradeLevel and **GradYear** for staff is always replaced with "staff".
GradeLevel and **GradYear** for employees is always replaced with "employee".
If street number comes from the StudentAddress/Address/Street/Line1 SIF element instead of the StreetNumber SIF element it will have extra digits in those cases where the street name has digits. 19283 W 4th Street would have a StreetNumber of 192834 instead of the expected 19283.

School Information swap words

Using the school information swap words is useful when the agent is attached to SQL table that has multiple schools. The swap words can be used to identify which school a student is enrolled. When using the agent to query a SQL table there is no automatic way to populate the schoolInfo table.
LocalId is replaced with the school number supplied by SIF in LocalId
SchoolName is replaced with the school name supplied by SIF
StatePrId is replaced with the state's school number supplied by SIF
NCESId is replaced with the NCES number supplied by SIF
IdentificationInfo1 is replaced with text supplied by SIF for IdentificationInfo1
IdentificationInfo2 is replaced with text supplied by SIF for IdentificationInfo2
IdentificationInfo3 is replaced with text supplied by SIF for IdentificationInfo3

For these school information swap words to work the agent needs to have the schoolInfo table populated. The following swap words require a SQL JOIN using SchoolInfoRefID from the userInfo table to the schoolInfo table.

A record should be added for each school with a SchoolInfoRefId and a schoolName. This can be done in the agent using SQL INSERT commands or using the query tool normally used to access the database.

ForXs2gApps SOFTWARE LICENSE AGREEMENT

In consideration for your use of the software and any updates, customizations and/or enhancements, entitled ForXs2gApps Agent provided by Hay River Software LLC ("Licensor"), you ("User") agree to the following terms and conditions. If you do not agree to these terms, you may not install the software and you must return the package to your point of purchase immediately for a refund.

1. License. Licensor hereby grants the User a non-exclusive, non-transferable license to use the Software for personal use on one computer by User only. Licensor reserves the right at any time, without liability or prior notice, to change the features or characteristics of the Software, this Agreement, or the Software's documentation and related materials.

2. License Restrictions.
a. User acknowledges that the Software and its structure, organization, and source code constitute valuable trade secrets of Licensor. Accordingly, User agrees not to (i) copy, perform, distribute, modify, adapt, alter, translate, or create derivative works from the Software; (ii) merge the Software with other software; (iii) sublicense, lease, rent, or loan the Software to any third party; (iv) reverse engineer, decompile, disassemble, or otherwise attempt to derive the source code for the Software; or (v) otherwise use the Software except as expressly allowed in this Agreement.
b. User shall comply with all applicable export and import control laws and regulations in its use of the Software and, in particular, User shall not export or re-export the Software without all required United States and foreign government licenses. User understands that access and use of the Software from outside the United States may constitute export of technology and technical data which may implicate export regulations and/or require export license.
c. Licensor retains exclusive ownership of all worldwide copyrights, trade marks, service marks, trade secrets, patent rights, moral rights, property rights and all other industrial rights in the Software and documentation, including any derivative works, modification, updates, or enhancements. All rights in and to the Software not expressly granted to User in this Agreement are reserved by Licensor. Nothing in this Agreement shall be deemed to grant, by implication, estoppel or otherwise, a license under any of Licensor's existing or future patents.
d. If User is an employee, contractor or agent of the United States Government, the following provision applies. The Software and documentation are comprised of "commercial computer software" and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (SEPT 1995) and are provided to the Government (i) for acquisition by or on behalf of civilian agencies, consistent with the policy set forth in 48 C.F.R. 12.212; or (ii) for acquisition by or on behalf of units of the Department of Defense, consistent with the policies set forth in 48 C.F.R. 227.7202-1 (JUN 1995) and 227.7202-3 (JUN 1995). Unpublished rights reserved under the copyright laws of the United States.
e. User shall not use the Software in any way that violates any local, state, federal or law of other nations, including but not limited to the posting of information that may violate third party rights, that may defame a third party, that may be obscene or pornographic, that may harass or assault others, that may violate hacking or other computer crime regulations, etc. Licensor does not monitor or edit any transmissions, postings, routings or other materials which User may send, post, route, transmit or otherwise move through or with the Software.

3. WARRANTY DISCLAIMER. THE SOFTWARE IS PROVIDED "AS IS" WITHOUT ANY WARRANTY WHATSOEVER, INCLUDING BUT NOT LIMITED TO ANY FUNCTIONALITY OR ITS BEING VIRUS FREE. USER RECOGNIZES THAT THE AS IS CLAUSE OF THIS AGREEMENT IS AN IMPORTANT PART OF THE BASIS OF THIS AGREEMENT, WITHOUT WHICH LICENSOR WOULD NOT HAVE AGREED TO ENTER THIS AGREEMENT. LICENSOR AND THIRD PARTIES DISCLAIM ALL WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE SOFTWARE, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT REGARDING THE SOFTWARE SHALL BE DEEMED A WARRANTY FOR ANY PURPOSE OR GIVE RISE TO ANY LIABILITY OF THIRD PARTIES WHATSOEVER. USER ACKNOWLEDGES THAT HE OR SHE HAS RELIED ON NO WARRANTIES OR STATEMENTS OTHER THAN AS MAY BE SET FORTH HEREIN.

4. LIMITATION OF LIABILITY. LICENSOR SHALL NOT BE LIABLE TO USER OR ANY THIRD PARTY FOR ANY INCIDENTAL, INDIRECT, EXEMPLARY, SPECIAL OR CONSEQUENTIAL DAMAGES, UNDER ANY CIRCUMSTANCES, INCLUDING, BUT NOT LIMITED TO, LOST PROFITS, REVENUE OR SAVINGS, LOSS OF GOODWILL, OR THE LOSS OF USE OF ANY DATA, EVEN IF LICENSOR HAD BEEN ADVISED OF, KNEW, OR SHOULD HAVE KNOWN, OF THE POSSIBILITY THEREOF. UNDER NO CIRCUMSTANCES SHALL LICENSOR'S AGGREGATE CUMULATIVE LIABILITY HEREUNDER, WHETHER IN CONTRACT, TORT, OR OTHERWISE, EXCEED THE TOTAL AMOUNT OF FEES ACTUALLY

PAID BY USER UNDER THIS AGREEMENT. USER ACKNOWLEDGES THAT THE FEES PAID BY HIM OR HER REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT LICENSOR WOULD NOT ENTER INTO THIS AGREEMENT WITHOUT THESE LIMITATIONS ON ITS LIABILITY.

5. Indemnification. User shall defend, indemnify and hold harmless Licensor, its officers, directors contractors, agents and employees, from any and all claims or causes of action arising out of use of or related to the Software, and pay any and all damages and expenses (including but not limited to attorneys fees incurred by Licensor and/or third parties) in connection therewith. Licensor reserves the right, at it own expense, to assume the exclusive defense and control of any matter otherwise subject to indemnification by User, in which event User shall cooperate with the Licensor in asserting any available defenses.

6. Termination. This Agreement is effective unless terminated by Licensor at any time for any breach of this Agreement. User may terminate this Agreement at any time by destroying all copies of the Software in User's possession and deleting the Software from User's computer system and other storage media, or by returning all such copies to Licensor. This Agreement and User's right to use this Software automatically terminate if User breaches this Agreement.

7. Legal Compliance. Licensor may suspend or terminate use of Software and this Agreement immediately upon receipt of any notice which alleges that User has used the Software for any purpose that violates any local, state, federal or law of other nations, including but not limited to the posting of information that may violate third party rights, that may defame a third party, that may be obscene or pornographic, that may harass or assault others, that may violate hacking or other criminal regulations, etc. of its agents, officers, directors, contractors or employees. In such event, Licensor may disclose the User's identity and contact information, if requested by a government or law enforcement body, or as a result of a subpoena or other legal action, and Licensor shall not be liable for damages or results thereof and User agrees not to bring any action or claim against this Licensor for such disclosure.

8. Miscellaneous. Either party may assign this Agreement to any successor in interest who purchases or through change in control owns greater than fifty percent of the assets or equity of such entity and agrees in writing to be bound by the terms and conditions herein; any other assignment shall be void. This Agreement and any dispute arising hereunder shall be construed in accordance with the laws of the State of Wisconsin without regard to principles of conflict of laws. For the purpose of this Agreement, User consents to the personal jurisdiction and venue of the state and federal courts located in Wisconsin. If any provision of this Agreement is prohibited by law or held to be unenforceable, the remaining provisions hereof shall not be affected, and this Agreement shall continue in full force and effect as if such unenforceable provision had never constituted a part hereof, and the unenforceable provision shall be automatically amended to so as to best accomplish the objectives of such unenforceable provision within the limits of applicable law. This Agreement may be executed in counterparts, each of which shall be deemed an original but all of which together shall constitute the same instrument. Any waiver of a provision of this Agreement must be in writing and signed by the party to be charged. A valid waiver hereunder shall not be interpreted to be a waiver of that obligation in the future or any other obligation under this Agreement. This Agreement constitutes the entire agreement between the parties related to the subject matter hereof, supersedes any prior or contemporaneous agreement between the parties relating to the Software and shall not be changed except by written agreement signed by an officer of Licensor.