

txttools Address Book Middleware Implementation Guide

June 30th 2010
Kris Bloe
cbloe@txttools.co.uk
Document Version 1.3
Software Version 1.3

Intended Audience

This document describes the implementation of the txttools Address Book Middleware application which helps txttools customers to integrate their core data systems with txttools and seamlessly copy contact and group data from the customer's data source to the txttools address book. End users of the txttools system, those users who actually use txttools to send SMS messages to recipients should not need to know anything about the Address Book Middleware. This document is designed for customer IT technicians who wish to automate the process of uploading contact and group data from their data source to txttools so that end users do not need to perform this manually.

Introduction

txttools is a web based application which provides customers with the core functionality for sending SMS messages to recipients. Users login to an account on the application through a browser, manage an address book of recipients, formulate groups and send messages to those recipients who receive them as SMS messages on their mobile device. In order to send messages users must first upload recipient data into the address book. This is often achieved by exporting data from a core system database into a file which is then uploaded through the web application user interface. Exporting and uploading large data sets introduces various challenges which the Address Book Middleware solution addresses.

Exporting data from a database and importing into a web application can be a laborious task to perform regularly. When large data sets are imported into txttools the data may change before another upload occurs rendering the imported data redundant. Automating the upload process solves these problems by removing the need for database administrators to regularly export data and end users to perform the file import, this removes the possibility of human error. Automation reduces data redundancy and keeps the data on txttools in sync with the customer's core system database.

The aim of the middleware application is to reduce the amount of work required by the customer to integrate their data source with txttools and negate the need for end users of the txttools system to manage their address book themselves.

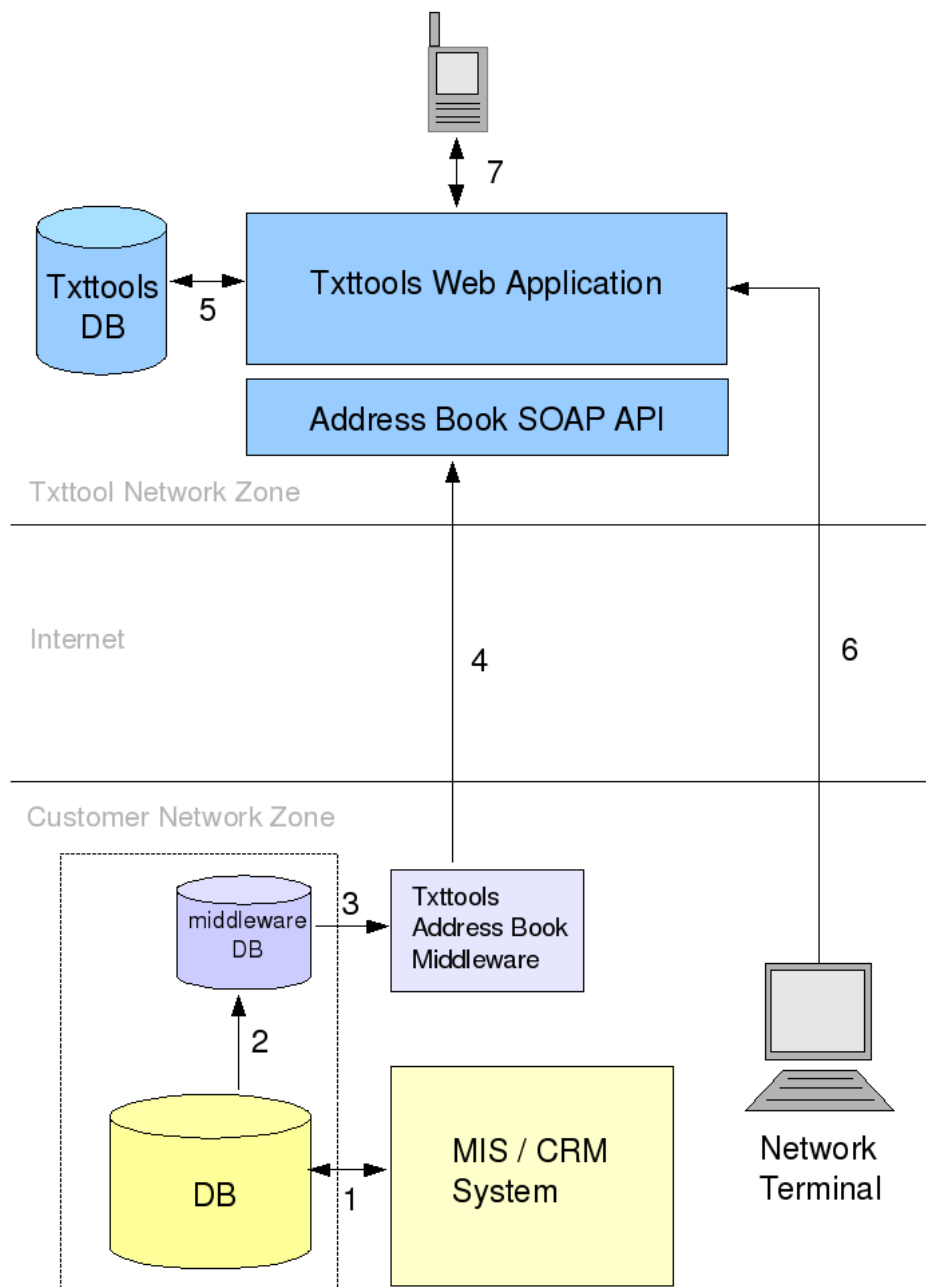
txttools provides a SOAP API through which users may upload address book data. The Address Book Middleware application makes use of this API and removes the need for users to create their own SOAP client.

Upgrading from an earlier version.

The only change required is a new column to the ACCOUNT_ADDRESSBOOK_OVERWRITE table. A new column KEEP_GROUPS varchar(5) will need to be added. This should either contain a value of 'TRUE' or 'FALSE' when entering rows into this table - setting this value to false will remove all groups as well as contacts when overwriting an address book.

Overview

txttools Address Book Middleware is a small Java application which takes contact and group data from a simple database and uploads it to txttools via the SOAP API. Users should populate that database with contact and group data from their own system which they wish to integrate. The middleware will run on any platform which hosts a Java Virtual Machine version 1.5 or above. The application can be configured to work with any database server, the middleware database can be hosted on any of the following SQL platforms: Oracle, MySQL or MS SQL Server. Creating the middleware schema and configuring the link between the middleware application and its database is covered in the later section of this document.



Data Flow

Data is moved between the customer's data source and txttools as follows, (See diagram above):

1. The customer's system records a new or changed record. This may be a change to a student's phone number, a change of course or removal from the system entirely. This data is recorded in the customer's system database.
2. A trigger on the relevant table in the customer's system updates the middleware dump database. The middleware database has three basic dump tables, One for contact information, one for group information and one for contact group membership. Contacts or groups can be flagged for removal, contact group memberships can be flagged as evictions.
3. The middleware system reads the dump database at regular intervals and formulates a SOAP request before removing the dump data after a successful SOAP invocation.
4. The middleware invokes the txttools SOAP Address Book API using securely stored txttools account credentials. The data is sent securely over Secure Socket Layer, (SSL), on port 443.
5. txttools imports the data from the request into the relevant account address book.
6. End users browse to www.txttools.co.uk, login to their account and can access the address book data according to their address book permission.

Users may perform the following txttools address book functions simply by populating the database with relevant data.

- Add/Delete a contact from an address book
- Update contact details already defined in an address book
- Add/Delete a group from an address book
- Update a group already defined in an address book
- Add new contacts as members of groups
- Move existing contacts into groups
- Evict existing group members from existing groups

Linking Middleware Database

The link between the middleware database and the customer's data source must be implemented by the customer. The database may be populated by any means, but the most obvious solution is to use a trigger on the core system database which copies the new, deleted or updated table row into the middleware database. When a contact's profile, or membership of a group is added, changed or deleted the trigger will take that information and use it to populate a stored procedure which runs on the middleware database. This is most effective if the middleware database is implemented inside the same database server as the core system database the customer wishes to integrate.

Customers who wish to implement the link between their system and the middleware database should ensure they have access to their database and that any such implementation of triggers and stored procedures or implementation of the middleware database within the same database server or physical machine does not adversely impact on the performance of their core system or commercial agreement with the provider of that system. txttools takes no responsibility for any effect the middleware may have on performance or commercial agreements with any third party. txttools cannot create this link for the customer as it will involve some kind of change to the customer's core system and txttools has no responsibility for changes to that system.

System Requirements

- Java SDK 1.5 or above
- Tomcat Servlet Container 5.5 or above
- HTTPS port 443 direct outbound access (will not work via a proxy)
- HTTP port 80 or 8080 in and out for initial configuration. This may be blocked once the middleware has been configured through the web interface.
- Database connectivity to which ever database server hosts the middleware database.

Implementation & Configuration

1. Ensure you have the Java JRE installed and the JAVA_HOME environment variable defined.
2. Install Tomcat and ensure the CATALINA_HOME environment variable is defined.
3. In \$CATALINA_HOME/conf/server.xml ensure there is a connector defined for the HTTP port. By default this is 8080. Ensure that any access restriction through the firewall allows HTTPS out from the server and HTTP in and out for set up purposes only. HTTP can be restricted later post configuration.
4. Now the tricky bit... The middleware application needs to be configured to connect to the middleware database. This is achieved by editing two xml files inside the war file.

If you are running Linux, you should be able to edit the contents of the war without unpacking it through an archive manager - if X is running. If running headless, perform a similar action to the Microsoft way.

If you're running Microsoft Windows, in order to change files inside the war file you will need to rename the .war to .zip, then unzip it to a directory. You can then go inside this directory and make the necessary changes detailed below, then re-zip the **contents** of the directory (not the directory itself), then change the extension back to .war. Once you have done this, keep this as a backup copy somewhere safe so you don't need to do this step again.

1. In /WEB-INF/applicationContext.xml comment out the inappropriate 'sessionFactory' and 'dataSource' databases, whilst uncommenting the required database details, ensuring the database ip/port and username/password are entered correctly.
 2. In /WEB-INF/hibernate.cfg.xml, uncomment the appropriate dialect.
5. Run the Middleware database script in which ever database server you have decided to use. Database scripts for Oracle, MySQL and MS SQL can be found in the war file itself in /WEB-INF/sql/. Note that for Oracle, the following changes will need to be made as required:
1. Line 18: CREATE TABLESPACE middleware DATAFILE
'/usr/local/oracle/oradata/middleware/middleware01.dbf' SIZE 10M AUTOEXTEND ON
NEXT 10M;
 2. Line 26: (username and db name should be the same as set in applicationContext.xml)

```
CREATE USER middleware IDENTIFIED BY t6KmF7aD DEFAULT TABLESPACE
middleware TEMPORARY TABLESPACE temp;
```

3. Lines 157-171: Change the database name 'mis' to the name of your MIS database.
6. Copy the Ttxttools_Addressbook_Middleware_1.3.war file into the \$CATALINA_HOME/webapps directory and boot tomcat. As long as the virtual host defined in server.xml has set unpackWARs="true", which it does by default, then tomcat will deploy the war file. Tail the tomcat logs (tail -f \$CATALINA_HOME/logs/catalina.out) and you should see the following INFO lines as the application boots.

```
Booting up the Addressbook Middleware service.....
##### Contact read timer is firing #####
```

then 30 seconds later -

```
##### Group read timer is firing #####
```

The contact timer and group timer will then run independently every 60 seconds.

7. Once the application is running you may use the web UI to add the txttools accounts which you wish to upload to. You will need to know the txttools username and password for each account. Depending upon where you are accessing the middleware UI from and which port you have set it to run on you'll need to open a browser. In this case I assume it is on local host running on port 8080 which is the default as described in point 3. Open a browser and got to http://localhost:8080/Txttools_Addressbook_Middleware_1.3/. You will see a very basic html page with a button 'add User'. Hit the button and add the txttools username and password for the account to which you want to upload the address book data. The credentials are stored in the middleware database, the passwords are secured using a password based encryption algorithm and are decrypted by the application when used to formulate the SOAP header. You may add as many accounts as you wish to the application, once you move back to the index page you will see the account usernames listed. Note that because the passwords are encrypted, you must enter user details via this interface – passwords cannot be entered directly into the database. If you wish to delete an account from the middleware application simply run an SQL delete statement on the USERS table for the relevant account.
8. You may now populate the middleware database with contact and group data. You should try running some SQL inserts into the tables to test the system and ensure the data is reflected on the designated txttools address book. The next section details the middleware database and how it should be populated.

Populating the Middleware Database

The middleware database comprises of 3 main tables for copying contact and group data.

- ACCOUNT_CONTACT holds contact details, names, phone numbers and all other txttools contact fields.
- ACCOUNT_GROUP holds the name and description for group data.
- ACCOUNT_CONTACT_GROUP relates contacts to group, either as members or for eviction.
- ACCOUNT_ADDRESSBOOK_OVERWRITE holds the account username for the address book which should be completely dropped at the next data upload.

All table have two common columns, ACCOUNT and ACTION. ACCOUNT defines the txttools account username the data relates to, i.e. the name of the txttools account which holds the address book the contact or group data is to be added to, updated in or deleted from. The system will match this value to the username in the USERS table and use that user to formulate the SOAP header when it makes the request. If no matching username is found in the USERS table the system will ignore that data row.

ACTION describes the action the SOAP request will make on the data, this can be either SAVE or DELETE. When adding new contact or group details, or updating existing entities in the address book this value should always be SAVE. This column will default to SAVE if omitted from an SQL insert. When adding a contact to a group, i.e. inserting a row into the ACCOUNT_CONTACT_GROUP table, the action should be SAVE, if you wish to evict a contact from a group the action should be DELETE.

Contacts should be uniquely identified by the UNIQUE_ID field, this value should be the unique identifier from your system. When referring to the contact again, to delete it, or manage it's group membership you should use this value. e.g. The CONTACT_ID column in ACCOUNT_CONTACT_GROUP should be the same value as the UNIQUE_ID you specify in the ACCOUNT_CONTACT table. There are no foreign keys in the middleware database, there is no defined relationship between ACCOUNT_CONTACT and ACCOUNT_CONTACT_GROUP or ACCOUNT_GROUP. You do not need to populate the ACCOUNT_CONTACT table in order to manage the group membership of a contact already defined in the txttools address book. The CONTACT_ID, i.e. the UNIQUE_ID of the contact you specified when uploading the contact data will identify that contact within the txttools address book. Similarly, the NAME value of the group identifies the group to which the contact should be added or removed. Groups are unique by name, case insensitive, within the txttools address book.

Omitting field values from the ACCOUNT_CONTACT or the DESCRIPTION from the ACCOUNT_GROUP will not result in that data being overwritten with a blank value in the txttools address book. If you wish to remove a field value from a contact then the field value must be specifically flagged to do so using the null flag value - #NULL#

For example, if you wish to change the phone number of a contact already defined within the txttools address book you simply need to provide that contact's unique id and the new phone number value. Any other existing fields on txttools will not be overwritten when the middleware updates that contact. If you wish to remove a value from a contact, such as the email address you again just need to specify the contact's unique id and set the EMAIL column value to #NULL#.

If you wish to drop the entire txttools address book before uploading another block of data you should enter that account username into the `ACCOUNT_ADDRESSBOOK_OVERWRITE` table. The next contact or group data which is uploaded will overwrite the entire txttools address book for that account. All subsequent uploads will update. So, if you wish to drop the entire address book before uploading another set of data simply enter the account name here before adding more data to the other tables. The `KEEP_GROUPS` column should either be `TRUE` or `FALSE` – setting this value to false will remove all groups as well as contacts.

Limitations

The middleware system works on a timer basis. The system will select all data from the middleware database for each account at a given period, by default this is every 60 seconds. The system will select a maximum number of rows from the ACCOUNT_CONTACT, ACCOUNT_CONTACT_GROUP and ACCOUNT_GROUP tables for each USER value, formulate the XML and make the SOAP request before deleting that data from the database. The system will only remove that data once a successful response has been received from the SOAP API. The default maximum is 100 rows per request. It is possible to populate the middleware database with huge data sets but users should be aware that the system is limited to 100 rows per request and that clearing the middleware database may take some time for large volumes of data.

It is advisable that the initial population of the txttools address book be made either by populating the middleware database or by a file upload before implementing any live link such as a trigger between the middleware database and the customer's core system datasource. This will prevent any lag between populating the middleware database and updating the txttools address book.

Further Assistance

If you require any help implementing or configuring the middleware application please contact the txttools technical team.

+44 (0)113 234 2111
techies@txttools.co.uk