

txttools Address Book Middleware Implementation Guide

October 12th 2011

Kris Bloe

techies@txttools.co.uk

Document Version 1.8

Software Version 1.3

Intended Audience

This document is designed for customer IT technicians who need to automate the uploading of contact and group data from their institution's MIS/data source to txttools.

Introduction

In order for txttools users to send SMS messages, recipient data must be added to their txttools address book. Historically, this was achieved by manually exporting MIS data into a csv file then uploaded via a browser, however, when MIS data is changed the txttools address book data is incorrect until the next csv upload.

The middleware application solves this problem.

Upgrading from an earlier version.

On upgrading to v1.3 the only change required is a new column to the ACCOUNT_ADDRESSBOOK_OVERWRITE table - a new column KEEP_GROUPS varchar(5) needs adding. This column requires 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, true will keep the groups.

Overview

The 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 txttools SOAP Address Book API. Users should populate that database with contact and group data from their own system which they wish to copy.

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 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 e.g. a change to a student's phone number, or a change of course or removal from the system entirely.
2. A trigger on the relevant table in the customer's system updates the middleware database. The middleware database contains three basic tables - one for contact information, one for group information and one for contact group membership. Contacts can be flagged for an update or removal and contact group memberships can be flagged as entries or evictions.
3. The middleware system reads the middleware database at regular intervals and formulates a SOAP request containing up to a given maximum [default] 100 rows.
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 uses the request data to update the relevant txttools address book, based upon the username field.
6. On a successful transmission, the data is removed from the middleware database.
7. 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 existing contacts
- Add/Delete a group from an address book
- Update existing groups
- Add new/existing contacts to new/existing groups
- Evict existing contacts 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 middleware database may be populated by any means, but the most obvious solution is to implement triggers on the core system database which updates the middleware database when a relevant change occurs.

It is advisable that the initial population of the txttools address book be made either by fully populating the middleware database or by a file upload before implementing any triggers. This will ensure the txttools address book is initially full.

Customers wishing 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 5+.
- Tomcat Servlet Container 5.5 or above (preferably 7).
- HTTP(S) direct outbound access (will not work via a proxy).
- HTTP port (default 8080) access for initial configuration. This may be blocked once the middleware has been configured through the web interface.
- Database connectivity to the database server hosting the middleware database.

Implementation & Configuration

1. Ensure you have the [Java \[6\] JRE](#) installed.
2. Install [Tomcat 7](#). If using Windows, opt for the Windows Service Installer and set Tomcat to boot on machine startup during the Tomcat install process..
3. Download the [latest](#) Middleware application.
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. Due to the UNIX file endings, the files are best opened in WordPad:

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 if it hasn't already been started. 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 in Linux or for Windows open the file named ...stdout.<date> in the logs directory of the Tomcat install) 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 independantly every 60 seconds, however at the bottom of the applicationContext.xml file, there's a bean created called timerController with constructor arguments of 60, 60 and 30. These figures denote the time between runs i.e. the Contact timer fires every 60 seconds, as does the Group timer and there's a stagger of 30 seconds between them.

If you wanted to increase the runs to every 5 minutes, for example, set these values to 300, 300 and 150. This will cause the Group timer to run 2.5 minutes after the Contact timer.

7. Once the application is running, visit (in a browser) the URL - http://localhost:8080/Ttxttools_Addressbook_Middleware_1.3/ - where you can add txttools accounts which you wish to upload address book data to. You will need to know the txttools username and password for each account. The credentials are stored in the middleware database, the passwords are secured using a two-way hash. 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 and that the data is subsequently removed from the middleware tables. The next section details the middleware database and how it should be populated.

Populating the Middleware Database

The middleware database comprises of 4 main tables for copying contact and group data.

- ACCOUNT_CONTACT holds contact details, names, phone numbers and all other 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.

The first three tables have two common columns, ACCOUNT and ACTION.

ACCOUNT - a txttools username which exists in the USERS table. If no matching username is found in the USERS table the system will ignore that data row.

ACTION - either SAVE (to add a new contact/group or update an existing contact/group) or DELETE. This column will default to SAVE if omitted from an SQL insert.

ID - any column called ID should not be assigned a value - the database will do this for you.

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 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 remove **all** existing contacts from the txttools address book for that account before the new data is added. 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/Known issues

The most common problems we find are to do with Gender and DOB entries: a contact's gender **must** be either "Male" or "Female" and the DOB **must** be of the form yyyy-mm-dd.

The USERS table cannot be populated by SQL - this has to be done via a browser.

MySQL Examples

Lets say for example you have a username in this table called 'user1'.

To add a new contact with just a few bits of data:

```
insert into ACCOUNT_CONTACT(UNIQUE_ID, ACCOUNT, NOTES,
PHONE_NUMBER) values("uid", "user1", "These are some
notes", "+447777777777");
```

To add a new full contact:

```
insert into ACCOUNT_CONTACT(UNIQUE_ID, ACCOUNT, FIRSTNAME,
LASTNAME, EMAIL, NOTES, PHONE_NUMBER, GENDER, HOUSE_NUMBER,
ADDRESS1, ADDRESS2, CITY, COUNTY, POSTCODE, COUNTRY, DOB)
values("uid", "user1", "Kris", "Bloe", "cbloe@txttools.co.uk", "These are some
notes", "+447777777777", "Male", "30", "Dock Street", "-", "Leeds", "West
Yorkshire", "LS10 1JF", "England", "1986-10-18");
```

To edit the contact - set the email to blank and update the notes:

```
insert into ACCOUNT_CONTACT(UNIQUE_ID, ACCOUNT, EMAIL, NOTES)
values("uid", "user1", "%NULL%", "New notes go here");
```

To delete the contact:

```
insert into ACCOUNT_CONTACT(UNIQUE_ID, ACCOUNT, ACTION)
values("uid", "user1", "DELETE");
```

To create a new group:

```
insert into ACCOUNT_GROUP(GROUP_NAME, DESCRIPTION, ACCOUNT)
values("Group1", "Description", "user1");
```

To edit the group - change the description:

```
insert into ACCOUNT_GROUP(GROUP_NAME, DESCRIPTION, ACCOUNT)
values("Group1", "New Description", "user1");
```

To edit the group - remove the description:

```
insert into ACCOUNT_GROUP(GROUP_NAME, DESCRIPTION, ACCOUNT)
values("Group1", "%NULL%", "user1");
```

To delete the group:

```
insert into ACCOUNT_GROUP(GROUP_NAME, ACCOUNT, ACTION)
values("Group1", "user1", "DELETE");
```

To add a contact to a group:

```
insert into ACCOUNT_CONTACT_GROUP(CONTACT_
ID, GROUP_NAME, ACCOUNT) values("uid", "Group1", "user1");
```

To evict the contact from the group:

```
insert into ACCOUNT_CONTACT_GROUP(CONTACT_ID, GROUP_NAME,
ACCOUNT, ACTION) values("uid", "Group1", "user1", "DELETE");
```

To delete all contacts and groups:

```
insert into ACCOUNT_ADDRESSBOOK_OVERWRITE(
ACCOUNT, KEEP_GROUPS) values("user1", "false");
```

To delete all contacts but keep any groups:

```
insert into ACCOUNT_ADDRESSBOOK_OVERWRITE(ACCOUNT, KEEP_GROUPS)
values("user1", "true");
```

NOTE - this will only run if an entry exists for the given username in any of the other three tables. So, if you wanted to clear the entire address book and leave it blank, you would need to run two SQL statements something similar to the following:

```
insert into ACCOUNT_ADDRESSBOOK_OVERWRITE(ACCOUNT, KEEP_GROUPS)
values("user1", "false");
```

```
insert into ACCOUNT_GROUP(GROUP_NAME, ACCOUNT) values("Empty
group", "user1");
```

This will remove everything, but leave one empty group.

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

MySQL

USE addressbook_middleware;

#txttools account credentials

```
DROP TABLE IF EXISTS `USERS` ;
CREATE TABLE `USERS` (
  `ID` int(6) NOT NULL AUTO_INCREMENT,
  `USERNAME` VARCHAR(100) NOT NULL,
  `PASSWORD` VARCHAR(255) NOT NULL,
  PRIMARY KEY(`ID`)
);
```

Contact data

```
DROP TABLE IF EXISTS `ACCOUNT_CONTACT` ;
CREATE TABLE `ACCOUNT_CONTACT` (
  `ID` int(10) NOT NULL AUTO_INCREMENT,
  `UNIQUE_ID` VARCHAR(100) NOT NULL,
  `ACCOUNT` VARCHAR(100) NOT NULL,
  `ACTION` VARCHAR(100) NOT NULL DEFAULT 'SAVE',
  `FIRSTNAME` VARCHAR(100),
  `LASTNAME` VARCHAR(100),
  `EMAIL` VARCHAR(100),
  `NOTES` VARCHAR(100),
  `PHONE_NUMBER` VARCHAR(100),
  `GENDER` VARCHAR(100),
  `HOUSE_NUMBER` VARCHAR(100),
  `ADDRESS1` VARCHAR(100),
  `ADDRESS2` VARCHAR(100),
  `CITY` VARCHAR(100),
  `COUNTY` VARCHAR(100),
  `POSTCODE` VARCHAR(100),
  `COUNTRY` VARCHAR(100),
  `DOB` VARCHAR(100),
  PRIMARY KEY (`ID`)
);
```

Contact group membership/eviction

```
DROP TABLE IF EXISTS `ACCOUNT_CONTACT_GROUP` ;
CREATE TABLE `ACCOUNT_CONTACT_GROUP` (
  `ID` int(10) NOT NULL AUTO_INCREMENT,
  `CONTACT_ID` VARCHAR(100) NOT NULL,
  `GROUP_NAME` VARCHAR(100) NOT NULL,
  `ACCOUNT` VARCHAR(100) NOT NULL,
  `ACTION` VARCHAR(100) NOT NULL DEFAULT 'SAVE',
  PRIMARY KEY (`ID`)
);
```

Group data

```
DROP TABLE IF EXISTS ACCOUNT_GROUP;
CREATE TABLE `ACCOUNT_GROUP` (
  `ID` int(10) NOT NULL AUTO_INCREMENT,
  `GROUP_NAME` VARCHAR(100) NOT NULL,
  `DESCRIPTION` VARCHAR(100),
  `ACCOUNT` VARCHAR(100) NOT NULL,
  `ACTION` VARCHAR(100) NOT NULL DEFAULT 'SAVE',
  PRIMARY KEY (`ID`)
);
```

```
# Address Book Overwrite
DROP TABLE IF EXISTS ACCOUNT_ADDRESSBOOK_OVERWRITE;
CREATE TABLE `ACCOUNT_ADDRESSBOOK_OVERWRITE` (
  `ID` int(10) NOT NULL AUTO_INCREMENT,
  `ACCOUNT` VARCHAR(100) NOT NULL,
  `KEEP_GROUPS` VARCHAR(5),
  PRIMARY KEY (`ID`)
);
```

MS SQL

```
/****** Object: Table [dbo].[USERS] Script Date: 04/20/2009 13:47:28
*****/
```

```
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[USERS]') AND type in (N'U'))
DROP TABLE [dbo].[USERS]
GO
```

```
/****** Object: Table [dbo].[ACCOUNT_ADDRESSBOOK_OVERWRITE] Script
Date: 06/17/2009 15:36:22 *****/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ACCOUNT_ADDRESSBOOK_OVERWRITE](
[ID] [int] IDENTITY(1,1) NOT NULL,
[ACCOUNT] [varchar](100) NOT NULL,
[KEEP_GROUPS] [varchar](100) NULL
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
```

```
/****** Object: Table [dbo].[USERS] Script Date: 04/20/2009 13:47:28
*****/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```

GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[USERS](
[ID] [int] NOT NULL IDENTITY(1,1),
[USERNAME] [varchar](100) NOT NULL,
[PASSWORD] [varchar](255) NOT NULL,
CONSTRAINT [UserID] PRIMARY KEY CLUSTERED
(
[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO

```

```

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
OBJECT_ID(N'[DF_ACCOUNT_CONTACT_ACTION]') AND type = 'D')
BEGIN
ALTER TABLE [dbo].[ACCOUNT_CONTACT] DROP CONSTRAINT
[DF_ACCOUNT_CONTACT_ACTION]
END
GO
/***** Object: Table [dbo].[ACCOUNT_CONTACT] Script Date: 04/20/2009
14:08:32 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ACCOUNT_CONTACT]') AND type in (N'U'))
DROP TABLE [dbo].[ACCOUNT_CONTACT]
GO
/***** Object: Table [dbo].[ACCOUNT_CONTACT] Script Date: 04/20/2009
14:08:32 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ACCOUNT_CONTACT](
[ID] [int] NOT NULL IDENTITY(1,1),
[UNIQUE_ID] [varchar](100) NOT NULL,
[ACCOUNT] [varchar](100) NOT NULL,
[ACTION] [varchar](100) NOT NULL,
[FIRSTNAME] [varchar](100) NULL,
[LASTNAME] [varchar](100) NULL,

```

```

[EMAIL] [varchar](100) NULL,
[NOTES] [varchar](100) NULL,
[PHONE_NUMBER] [varchar](100) NOT NULL,
[GENDER] [varchar](10) NULL,
[HOUSE_NUMBER] [varchar](10) NULL,
[ADDRESS1] [varchar](100) NULL,
[ADDRESS2] [varchar](100) NULL,
[CITY] [varchar](100) NULL,
[COUNTY] [varchar](100) NULL,
[POSTCODE] [varchar](15) NULL,
[COUNTRY] [varchar](100) NULL,
[DOB] [varchar](20) NULL,
CONSTRAINT [PK_ACCOUNT_CONTACT] PRIMARY KEY CLUSTERED
(
[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[ACCOUNT_CONTACT] ADD CONSTRAINT
[DF_ACCOUNT_CONTACT_ACTION] DEFAULT ('SAVE') FOR [ACTION]
GO

```

```

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
OBJECT_ID(N'[DF_ACCOUNT_CONTACT_GROUP_ACTION]') AND type = 'D')
BEGIN
ALTER TABLE [dbo].[ACCOUNT_CONTACT_GROUP] DROP CONSTRAINT
[DF_ACCOUNT_CONTACT_GROUP_ACTION]
END
GO
/***** Object: Table [dbo].[ACCOUNT_CONTACT_GROUP] Script Date: 04/
20/2009 14:10:29 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ACCOUNT_CONTACT_GROUP]') AND type in (N'U'))
DROP TABLE [dbo].[ACCOUNT_CONTACT_GROUP]
GO
/***** Object: Table [dbo].[ACCOUNT_CONTACT_GROUP] Script Date: 04/
20/2009 14:10:29 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON

```

```

GO
CREATE TABLE [dbo].[ACCOUNT_CONTACT_GROUP](
[ID] [int] NOT NULL IDENTITY(1,1),
[CONTACT_ID] [varchar](100) NOT NULL,
[GROUP_NAME] [varchar](100) NOT NULL,
[ACCOUNT] [varchar](100) NOT NULL,
[ACTION] [varchar](100) NOT NULL,
CONSTRAINT [PK_ACCOUNT_CONTACT_GROUP] PRIMARY KEY CLUSTERED
(
[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[ACCOUNT_CONTACT_GROUP] ADD CONSTRAINT
[DF_ACCOUNT_CONTACT_GROUP_ACTION] DEFAULT ('SAVE') FOR [ACTION]
GO

```

```

IF EXISTS (SELECT * FROM dbo.sysobjects WHERE id =
OBJECT_ID(N'[DF_ACCOUNT_GROUP_ACTION]') AND type = 'D')
BEGIN
ALTER TABLE [dbo].[ACCOUNT_GROUP] DROP CONSTRAINT
[DF_ACCOUNT_GROUP_ACTION]
END
GO
/***** Object: Table [dbo].[ACCOUNT_GROUP] Script Date: 04/20/2009
14:11:14 *****/
IF EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ACCOUNT_GROUP]') AND type in (N'U'))
DROP TABLE [dbo].[ACCOUNT_GROUP]
GO
/***** Object: Table [dbo].[ACCOUNT_GROUP] Script Date: 04/20/2009
14:11:14 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[ACCOUNT_GROUP](
[ID] [int] NOT NULL IDENTITY(1,1),
[GROUP_NAME] [varchar](100) NOT NULL,
[DESCRIPTION] [varchar](100) NOT NULL,

```

```

[ACCOUNT] [varchar](100) NOT NULL,
[ACTION] [varchar](100) NOT NULL,
CONSTRAINT [PK_ACCOUNT_GROUP] PRIMARY KEY CLUSTERED
(
[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
ALTER TABLE [dbo].[ACCOUNT_GROUP] ADD CONSTRAINT
[DF_ACCOUNT_GROUP_ACTION] DEFAULT ('SAVE') FOR [ACTION]
GO

```

ORACLE

```

-- Ian Hudson
-- ihudson@txttools.co.uk
-- Middleware Oracle database script v1.0.0.
-- 20/04/2009.
-- 29/04/2009 - Rename table USER to USERS (Oracle keyword).
-- 17/06/2009 - Additional table ACCOUNT_ADDRESSBOOK_OVERWRITE

```

```

-- Create middleware schema.

```

```

-----
DROP USER middleware CASCADE;
DROP TABLESPACE middleware INCLUDING CONTENTS AND DATAFILES;

```

```

-----
-- Edit directory path to reflect the path of your Oracle instance dbf directory.
-----

```

```

CREATE TABLESPACE middleware DATAFILE '/usr/local/oracle/oradata/
middleware/middleware01.dbf' SIZE 10M AUTOEXTEND ON NEXT 10M;
-----

```

```

-----
-- Any password change here needs to be reflected in /WEB-INF/
ApplicationContext.xml.
-----

```

```

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

```

```

ALTER USER middleware QUOTA unlimited ON middleware;
GRANT CREATE SESSION TO middleware;
GRANT CREATE TABLE TO middleware;
GRANT CREATE VIEW TO middleware;
GRANT CREATE SEQUENCE TO middleware;
GRANT CREATE PROCEDURE TO middleware;
GRANT CREATE TRIGGER TO middleware;
GRANT SELECT ANY TABLE to middleware;
commit;
-----
-- Create middleware tables.
-- Create middleware auto increment sequences.
-- Create middleware auto increment triggers.
-----
-----
--Table ACCOUNT_CONTACT
-----
CREATE TABLE "MIDDLEWARE"."ACCOUNT_CONTACT"
( "ID" NUMBER(11,0),
"UNIQUE_ID" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"ACCOUNT" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"ACTION" VARCHAR2(100 BYTE) DEFAULT 'SAVE' NOT NULL ENABLE,
"FIRSTNAME" VARCHAR2(100 BYTE),
"LASTNAME" VARCHAR2(100 BYTE),
"EMAIL" VARCHAR2(100 BYTE),
"NOTES" VARCHAR2(100 BYTE),
"PHONE_NUMBER" VARCHAR2(100 BYTE),
"GENDER" VARCHAR2(100 BYTE),
"HOUSE_NUMBER" VARCHAR2(100 BYTE),
"ADDRESS1" VARCHAR2(100 BYTE),
"ADDRESS2" VARCHAR2(100 BYTE),
"CITY" VARCHAR2(100 BYTE),
"COUNTY" VARCHAR2(100 BYTE),
"POSTCODE" VARCHAR2(100 BYTE),
"COUNTRY" VARCHAR2(100 BYTE),
"DOB" VARCHAR2(100 BYTE),
CONSTRAINT "PRIMARY00000" PRIMARY KEY ("ID")
);
DROP SEQUENCE MIDDLEWARE.ACCOUNT_CONTACT_0;
CREATE SEQUENCE "MIDDLEWARE"."ACCOUNT_CONTACT_0" MINVALUE 1
MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1 START WITH
1 NOCACHE ORDER NOCYCLE;
CREATE OR REPLACE TRIGGER "MIDDLEWARE"."ACCOUNT_CONTACT" BEFORE
INSERT ON "MIDDLEWARE"."ACCOUNT_CONTACT" FOR EACH ROW WHEN
(new."ID" is null) begin select "ACCOUNT_CONTACT_0".nextval into :new."ID"
from dual; end;
/
SHOW ERRORS;
ALTER TRIGGER "MIDDLEWARE"."ACCOUNT_CONTACT" ENABLE;

```

--Table ACCOUNT_CONTACT_GROUP

```
CREATE TABLE "MIDDLEWARE"."ACCOUNT_CONTACT_GROUP"
( "ID" NUMBER(11,0),
"CONTACT_ID" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"GROUP_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"ACCOUNT" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"ACTION" VARCHAR2(100 BYTE) DEFAULT 'SAVE' NOT NULL ENABLE,
CONSTRAINT "PRIMARY00001" PRIMARY KEY ("ID")
);
DROP SEQUENCE "MIDDLEWARE"."ACCOUNT_CONTACT_GROUP_0";
CREATE SEQUENCE "MIDDLEWARE"."ACCOUNT_CONTACT_GROUP_0"
MINVALUE 1 MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1
START WITH 1 NOCACHE ORDER NOCYCLE ;
CREATE OR REPLACE TRIGGER "MIDDLEWARE"."ACCOUNT_CONTACT_GROUP"
BEFORE INSERT ON "MIDDLEWARE"."ACCOUNT_CONTACT_GROUP"
FOR EACH ROW WHEN (new."ID" is null) begin
select "ACCOUNT_CONTACT_GROUP_0".nextval into :new."ID" from dual; end;
/
SHOW ERRORS;
ALTER TRIGGER "MIDDLEWARE"."ACCOUNT_CONTACT_GROUP" ENABLE;
```

--Table ACCOUNT_GROUP

```
CREATE TABLE "MIDDLEWARE"."ACCOUNT_GROUP"
( "ID" NUMBER(11,0),
"GROUP_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"DESCRIPTION" VARCHAR2(100 BYTE),
"ACCOUNT" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"ACTION" VARCHAR2(100 BYTE) DEFAULT 'SAVE' NOT NULL ENABLE,
CONSTRAINT "PRIMARY00002" PRIMARY KEY ("ID")
);
DROP SEQUENCE "MIDDLEWARE"."ACCOUNT_GROUP_0";
CREATE SEQUENCE "MIDDLEWARE"."ACCOUNT_GROUP_0" MINVALUE 1
MAXVALUE 99999999999999999999999999999999 INCREMENT BY 1 START WITH
1 NOCACHE ORDER NOCYCLE ;
CREATE OR REPLACE TRIGGER "MIDDLEWARE"."ACCOUNT_GROUP" BEFORE
INSERT ON "MIDDLEWARE"."ACCOUNT_GROUP" FOR EACH ROW WHEN
(new."ID" is null) begin select "ACCOUNT_GROUP_0".nextval into :new."ID"
from dual; end;
/
SHOW ERRORS;
ALTER TRIGGER "MIDDLEWARE"."ACCOUNT_GROUP" ENABLE;
```

```

-----
-- Table USERS
-----
CREATE TABLE "MIDDLEWARE"."USERS"
( "ID" NUMBER(11,0),
"USERNAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"PASSWORD" VARCHAR2(255 BYTE) NOT NULL ENABLE,
CONSTRAINT "PRIMARY00003" PRIMARY KEY ("ID")
);
DROP SEQUENCE "MIDDLEWARE"."USER_0";
CREATE SEQUENCE "MIDDLEWARE"."USER_0" MINVALUE 1 MAXVALUE
99999999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE
ORDER NOCYCLE ;
CREATE OR REPLACE TRIGGER "MIDDLEWARE"."USERS" BEFORE INSERT
ON "MIDDLEWARE"."USERS" FOR EACH ROW WHEN (new."ID" is null) begin
select "USER_0".nextval into :new."ID" from dual; end;
/
SHOW ERRORS;
ALTER TRIGGER "MIDDLEWARE"."USERS" ENABLE;
-----
-- ACCOUNT_ADDRESSBOOK_OVERWRITE
-----
CREATE TABLE "MIDDLEWARE"."ACCOUNT_ADDRESSBOOK_OVERWRITE"
( "ID" NUMBER(11,0),
"ACCOUNT" VARCHAR2(100) NOT NULL ENABLE,
"KEEP_GROUPS" VARCHAR2(100),
CONSTRAINT "PRIMARY00004" PRIMARY KEY ("ID")
);
DROP SEQUENCE "MIDDLEWARE"."ACC_ADD_0";
CREATE SEQUENCE "MIDDLEWARE"."ACC_ADD_0" MINVALUE 1 MAXVALUE
99999999999999999999999999999999 INCREMENT BY 1 START WITH 1 NOCACHE
ORDER NOCYCLE ;
CREATE OR REPLACE TRIGGER "MIDDLEWARE"."ACC_ADD" BEFORE INSERT
ON "MIDDLEWARE"."ACCOUNT_ADDRESSBOOK_OVERWRITE" FOR EACH ROW
WHEN (new."ID" is null) begin select "ACC_ADD_0".nextval into :new."ID"
from dual; end;
/
SHOW ERRORS;
ALTER TRIGGER "MIDDLEWARE"."ACC_ADD" ENABLE;
-----
-- Here we grant the institutions MIS permissions
-- to insert update & delete on all the middleware
-- tables.
--
-- mis NEEDS TO BE EDITED TO REFLECT THE SCHEMA OWNER
-- OF THE MIS TABLE SET.
-----
grant insert on middleware."ACCOUNT_CONTACT" to mis;
grant update on middleware."ACCOUNT_CONTACT" to mis;

```

```
grant delete on middleware."ACCOUNT_CONTACT" to mis;  
grant insert on middleware."ACCOUNT_CONTACT_GROUP" to mis;  
grant update on middleware."ACCOUNT_CONTACT_GROUP" to mis;  
grant delete on middleware."ACCOUNT_CONTACT_GROUP" to mis;  
grant insert on middleware."ACCOUNT_GROUP" to mis;  
grant update on middleware."ACCOUNT_GROUP" to mis;  
grant delete on middleware."ACCOUNT_GROUP" to mis;  
grant insert on middleware."USERS" to mis;  
grant update on middleware."USERS" to mis;  
grant delete on middleware."USERS" to mis;  
commit;
```