User Manual for connecting to SQL Server from Linux and Mac Machines with Active Directory Authentication

Prepared by ITSD DBA Team (Anupinder Rai, Jeff Wagner and Leyla Rutz).

For any query please mail idba@stsci.edu

Table of Content

1) **Kerberos** **3**
    i) **Kinit**
    ii) **Kdestroy**
    iii) **Klist**
    iv) **Ktutil**
2) **UnixODBC** **4**
3) **SQL Tools** **5**
    i) **Sqlcmd**
    ii) **Sqsh**
    iii) **SQL squirrel**
4) **Access from programming languages** **8**
    i) **Java**
    ii) **Python**
    iii) **Perl**
5) **References** **11**

KERBEROS

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Kinit

kinit obtains and caches an initial ticket-granting ticket for *principal*.

Kdestroy

The kdestroy utility destroys the user's active Kerberos authorization tickets by overwriting and deleting the credentials cache that contains them. If the credentials cache is not specified, the default credentials cache is destroyed.

Klist

The klist command lists the Kerberos principal and Kerberos tickets held in a credentials cache, or the keys held in a keytab file.

Ktutil

The ktutil command invokes a subshell from which an administrator can read, write, or edit entries in a Kerberos file.

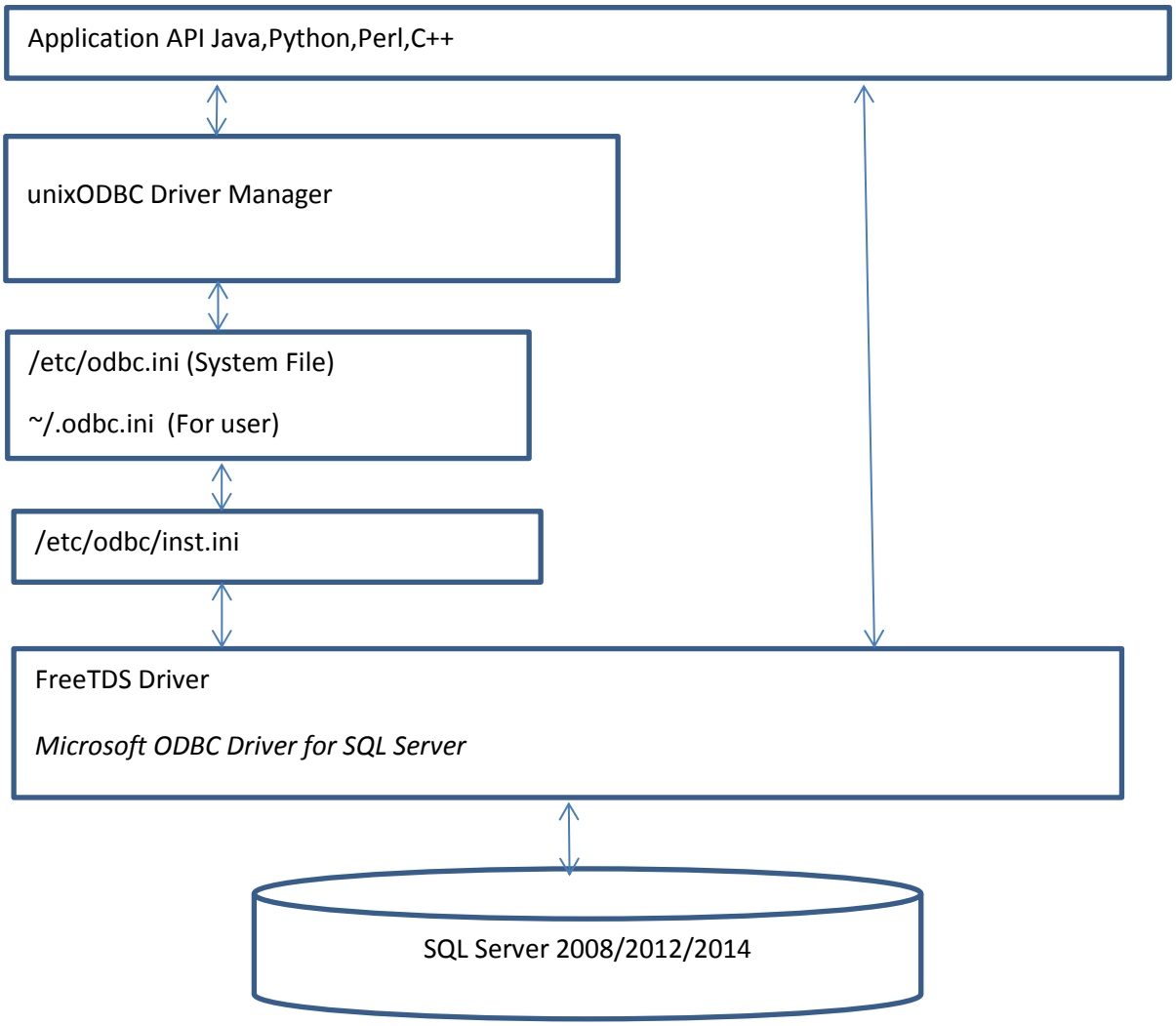
Here is the process to store credentials in the keytab file. This key tab file will be used in example programs

```
ktutil

addent -password -p <username> -k 2 -e rc4-hmac
wrt <filename>.keytab
exit
```

unixODBC Driver :

unixODBC is an open source project that implements the ODBC API. The code is provided under the GNU GPL/LGPL and can be built and used on many different operating systems, including most versions of Unix, Linux, Mac OS X.



odbc.ini

This file stores DSN (Data Source Name) information for DSNs created to connect to database servers. DSNs contain information like:

Server name

Port number

Driver

This is a sample DSN that will be used to connect to a database server:

```
[jwdmsdevdbvm1]
Driver = FreeTDS
Server = jwdmsdevdbvm1.stsci.edu
Port = 1433
TDS_Version = 7.1
Trusted_Connection = Yes
```

There is an odbc.ini located in the /etc directory which can be read by all users. Each user can create .odbc.ini in their home directory it is accessible by that user only.

odbcinst.ini

This file is in /etc directory and contains information about all drivers that are registered with unixODBC Driver manager. Example of drivers in odbcinst.ini are as follows

```
[FreeTDS]
Description=TDS Driver (MS SQL)
Driver=/usr/local/lib/libtdsodbc.so
UsageCount=1

[ODBC Driver 11 for SQL Server]
Description=Microsoft ODBC Driver 11 for SQL Server
Driver=/opt/microsoft/msodbcsql/lib64/libmsodbcsql-11.0.so.2270.0
Threading=1
UsageCount=1
```

tsql utility to test FreeTDS connections and queries

```
tsql –S<servername>
```

isql Utility to test DSN connection and queries

isql -S<DSN> -U<username> -P<password>

SQLCMD

The sqlcmd utility is available in the Microsoft ODBC Driver for SQL Server on Linux. Paramter E is used for Active Directory authentication. On linux machines, you are required to have a Kerberos ticket. It can be checked with the klist command and can be created with kinit command.

sqlcmd –S <servername> -E

sqsh (pronounced skwish) is short for SQshelL (pronounced s-q-shell). It is a tool to connect to a database server.

sqsh -S <servername> -U 'stsci\username'

It prompts for active directory password

SQL Squirrel

Squirrel is a graphical tool to connect to Microsoft SQL Server as well as other database servers.

The following are steps used to setup Active Directory Authentication to SQL Server for SQL Squirrel

- **Get Microsoft JDBC Driver version 4 driver from Microsoft site and unzip it.**
- **Copy sqljdbc4.jar file to Contents->Resources->Java->lib directory of SQL Squirrel application**
- **Open squirrel-sql.sh file and change**

```
MACOSX_SQUIRREL_PROPS="-Dapple.laf.useScreenMenuBar=true
-Dcom.apple.mrj.application.apple.menu.about.name=SQuirreLSQL"
```

to

```
MACOSX_SQUIRREL_PROPS="-Djava.security.krb5.realm=STSCI.EDU
-Djava.security.krb5.kdc=pwidm01.stsci.edu:88
-Dapple.laf.useScreenMenuBar=true
-Dcom.apple.mrj.application.apple.menu.about.name=SQuirreLSQL"
```

- **Start the SQL Squirrel Application and select Microsoft MSSQL Server JDBC Driver from the list of drivers under the Drivers tab and provide a name.**
- **Click on the "Aliases" tab. In the tray of icons click on the "+" icon to add a new alias. Use the popup menu in the new window to select from the list the driver you just created.**
- **Type the "Name" field to identify your alias.**
- **Edit URL to jdbc:sqlserver://<server_name>:1433;databaseName=<db_name>**
- **Do not enter User Name or Password , instead click on the "Properties" button.In the properties window, select the "Driver Properties" tab. In "Driver Properties" click on the "Use Driver Properties" checkbox in the upper left.**
- **Check the box for integratedSecurity and change its value to "true"**

- **Check the box for authenticationScheme and select javaKerberos for its value.**
- **Click the OK button**
- **Run a kinit in your Terminal.**
- **Restart SQuirreL , double click on your new Alias and hit the connect button.**

Java

To access SQL Server from a java program, Microsoft JDBC driver will be used. It will use JavaKerberos security and will connect to SQL Server directly.

Follow following steps

- **add location of Microsoft JDBC driver to CLASSPATH**
- **echo $CLASSPATH**
- **home/arai/sqljdbc_4.0/enu/sqljdbc4.jar**
- **Create a keytab file as mentioned in the instructions on page #**
- **Create configuration file that contains information about keytab file like SQLJDBCDriver.conf created here**

```
SQLJDBCDriver {
  com.sun.security.auth.module.Krb5LoginModule required principal=arai useKeyTab=true
keyTab="/home/arai/arai.keytab";
};
```

- **Create a Java program specifying Microsoft ODBC driver and authentication scheme as JavaKerberos**

```
Import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class test {

    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        Connection conn =
DriverManager.getConnection("jdbc:sqlserver://sqlmgtvm.stsci.edu;database=master;integratedSecurity=true;a
uthenticationScheme=JavaKerberos");
        Statement sta = conn.createStatement();
        String Sql = "select name from sys.databases";
        ResultSet rs = sta.executeQuery(Sql);
        while (rs.next()) {
            System.out.println(rs.getString("name"));
        }
    }
}
```

- **Compiling Java Program**

```
javac ./test.java
```

- **Interpreting (Compiling) Java program with configuration file specified above**

```
java test -Djava.security.auth.login.config=/home/arai/SQLJDBCDriver.conf
```

Python

Python programs connect to SQL Server with unixODBC driver Manager and at the driver layer with either FreeTDS or Microsoft ODBC driver. The program can make connections with either a DSN or a connection string specifying connection details. The following examples show how to use different methods. For Kerberos part, ticket can be generated either with keytab file or with kinit.

Program 1 using keytab file and DSN

```
import os
os.system('kinit arai -k -t /home/arai/key1.keytab')
import pyodbc
con = pyodbc.connect('DSN=CHICO')
cur = con.cursor()
cur.execute('SELECT @@servername')
print cur.fetchall()
```

Program 2 uses temporary Kerberos ticket created with kinit.It is must to create ticket with kinit for this program to work

```
import pyodbc
con = pyodbc.connect('DSN=CHICO')
cur = con.cursor()
cur.execute('SELECT @@servername')
print cur.fetchall()
```

Program 3 uses ketyab file with full connection string instead of DSN

```
os.system('kinit arai -k -t /home/arai/key1.keytab')
import pyodbc
con = pyodbc.connect('Driver=MS;Server=tcp:sqlmgtvm.stsci.edu,1433; Trusted_Connection=yes')
cur = con.cursor()
cur.execute('SELECT @@servername')
print cur.fetchall()
```

Perl

Perl program uses DBI::ODBC library to connect to SQL Server.

```
use DBI;
use strict;
my $dbh = DBI-> connect('dbi:ODBC:DSN=test0') or die "CONNECT ERROR! :: $DBI::err $DBI::errstr $DBI::state
$!\n";
if ($dbh)
{
 print "There is a Connection\n";
 my $sql = q/SELECT auth_scheme from sys.dm_exec_connections where session_id=@@spid/;
 my $sth = $dbh->prepare($sql);
 $sth->execute();
 my @row;
 while (@row = $sth->fetchrow_array) {
  print join(", ", @row), "\n";
 }
 $dbh->disconnect;
 }
```

References

http://web.mit.edu/kerberos/krb5-devel/doc/user/user_commands/klist.html

http://web.mit.edu/kerberos/krb5-1.12/doc/user/user_commands/kinit.html

http://www.unixodbc.org/