TD MOBILE™

Database Connectivity Guide

Product Version 2.0

GUPTA

TD Mobile™: Database Connectivity Guide, Product Version 2.0

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Last updated: September 10, 2015.

Legal Notice

THE DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. GUPTA TECHNOLOGIES, INC. SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

This document may describe features and/or functionality not present in your software or your service agreement. Contact your account representative to learn more about what is available with this Gupta Technologies® product.

Gupta Technologies, Inc.
1420 Rocky Ridge Drive, Suite 380
Roseville, CA 95661

Gupta Technologies.com

Table of Contents

Introduction

TD Mobile allows connecting to various backend databases natively including SQLBase, SQLServer and Oracle plus any ODBC or OLEDB compatible data sources.

It uses Gupta's existing TD.NET runtimes, including our database assembly (gupta.td.runtime.database.dll). This assembly handles all connects, disconnects, fetches, executes, bind processing, etc. that occur in an application. All database interaction is done in the same manner as in traditional Team Developer applications. For example, use the function SqlPrepareAndExecute to prepare and execute your Selects, Updates, Deletes, etc., SqlFetchNext to move through your result set and so on. One area of difference between TD and TD Mobile applications is how connections should be handled.

In traditional Team Developer applications, the system variables SqlDatabase, SqlUser, and SqlPassword are given values describing the connection and then the function SqlConnect is called to establish the connection. In TD.NET, these variables are implemented as “static” properties inside our underlying database class. In a TD Mobile environment, where a single copy of our database assembly is shared across all IIS connections, those properties are “static” across all the threads. This can result in multiple users setting those variables to new values all the time, possibly interfering with each other's changes.

In addition, our existing database connectivity has built-in logic for a kind of pooling or recycling of connections but this logic also relies on static structures. So, different threads (users) could be trying to recycle other user’s connections.

Although the traditional connection methods of Team Developer may seem to work fine, under multiple user loads unpredictable results may occur due to its stateless nature.

For this reason, we recommend using the TD.NET function "SqlConnectDotNet" to support connecting to databases with "connection strings". It is possible to connect to any database via OLEDB, ODBC or .Net Data Provider. The format is:

bOk = SqlConnectDotNet(hSql, sConnectionString, sInvariantName, nProviderType)

Parameters:

hSql - Receive Sql Handle. This is a handle that identifies a database connection.

sConnectionString - String. This is a .NET connection string that contains all the information needed to connect to the database, including the username and password.

sInvariantName - String. This is an ADO.NET invariant name, which should correspond to the invariant attribute of the factory entry in the section of your machine.config file, i.e. System.Data.SqlClient.

- Adaptive Server Anywhere: "iAnywhere.Data.SQLAnywhere"
- DB2: "IBM.Data.DB2"
- Firebird: "FirebirdSql.Data.FirebirdClient"
- Ingres: "Ingres.Client"
- MySQL: "MySql.Data.MySqlClient"
- Oracle: ODP.NET: "Oracle.DataAccess.Client"
- Oracle: MS Oracle: "System.Data.OracleClient"
- SQLite: System.Data.SQLite"
- SQL CE: v3.0: "System.Data.SqlServerCe"
- SQL CE: v3.5: "System.Data.SqlServerCe.3.5"
- Sybase ASE: "Sybase.Data.AseClient"

nProviderType - Number. This is a DBP_PROVIDER constant that identifies the database provider you want to use. This should be one of the following values.

- DBP_PROVIDER_UNDEFINED=0
- DBP_PROVIDER_SQLBASE_OLEDB=1
- DBP_PROVIDER_SQLSERVER_OLEDB=2
- DBP_PROVIDER_ORACLE_OLEDB=3
- DBP_PROVIDER_ODBC=4
- DBP_PROVIDER_ORACLE=5
- DBP_PROVIDER_OLEDB=7
- DBP_PROVIDER_SQLBASE=9
- DBP_PROVIDER_SQLSERVER_SQLCLIENT=12

Return Values:

bOk is TRUE if the function succeeds and FALSE if it fails.

This function allows connections to arbitrary databases using a .NET connection string. Unlike SqlConnect, this function does not rely on or check the value of SqlUser, SqlPassword, or SqlDatabase. Instead, all of the information needed to connect is stored in the connection string.

This function has several benefits:

- It doesn’t use the built-in connection pooling.
- It doesn’t get the connection information from SqlDatabase/SqlUser/SqlPassword (or even sql.ini) but instead just takes a standardized .NET connection string.

By using SqlConnectDotNet to connect to the database, the connection process becomes fully thread safe. Fortunately, connection pooling is already built into most database drivers and is turned on by default, so you still have access to the performance benefit of

ing. For SQLBase specifically, it is supported but ***off*** by default, so you need to add a tag to your connection string to turn pooling on.

PoolSize=<#ofConnections>;ConnectionLifeTime=<In Seconds>

Where “PoolSize” is the maximum size of the pool you want to maintain and “ConnectionLifeTime” is the longest amount of time you want an unused connection to remain in the pool before getting rid of it.

Care and thought must also be given to the scope of any Sql Handle variables your application may use. It is possible to use a global variable as a Sql Handle and then just leave it connected between Operation invocations. Keep in mind that the user may navigate away from your application and you'll have no way of recognizing that condition, so the database connection will needlessly remain until the IIS session times out and the variable is destroyed. It is considered better practice, when possible, to do a connect and disconnect for each operation and not maintain a database connection "state".

Connecting to SQLBase:

It is possible to connect to SQLBase via Dot Net Provider, ODBC or OLEDB. The function call to connect to SQLBase via .Net Provider will be similar to the following.

```
Call SqlConnectDotNet(hSql, 'server-
name=server1;hostname=localhost;port=2155;database=regis;user=sysadm;password=sysad
m;poolsize=10;connectionlifetime=20;',””, DBP_PROVIDER_SQLBASE )
```

Connecting to SQLServer:

Via OLEDB:

It is possible to connect to SQLServer via an OLEDB Provider. The function call will be similar to the following.

```
Call SqlConnectDotNet(hSqlConnect, 'Provider=sqloledb;Data
Source=KRAGAVAN15\\SQLEXPRESS;Initial Catalog=Test;User
Id=Test;Password=test;', "", DBP_PROVIDER_SQLSERVER_OLEDB )
```

Via ODBC:

It is possible to connect to SQLServer via ODBC as well. The function call will be similar to the following.

```
Call SqlConnectDotNet(hSqlConnect, 'Driver={SQL Server Native Client
11.0};Server=KRAGAVAN15\\SQLEXPRESS;Database=TEST;Uid=test;Pwd=test;', "",
DBP_PROVIDER_ODBC )
```

Connecting to Oracle:

Via .Net Data Provider:

Connection to Oracle can be established via .NetDataProvider, OLEDB or ODBC. To establish connectivity use a call similar to the following.

```
SqlConnectDotNet(hSqlConnect, 'Data Source=ORCLWin1252;User
Id=support;Password=support;',  "", DBP_PROVIDER_ORACLE )
```

Managed Data Access

TD Mobile uses Oracle.ManagedDataAccess.dll to connect to ORACLE.

Oracle.ManagedDataAccess.dll is a full .NET version and it does not require OCN libraries. You can install it by using ODP.NET, with either the 32- or 64-bit installer.

Note that it uses a different location for tnsnames.ora than does the standard ORACLE client, and therefore it may disable the TNS name setting.

Oracle.ManagedDataAccess.dll is also installed with the Oracle 12c client. If you install the client, then Oracle.ManagedDataAccess.dll shares the same tnsnames.ora with other Oracle client tools.

Samples:

Please refer to the samples installed by the TD Mobile setup in order to see how to use the connectivity functions described in this document.

Gupta Technologies
1420 Rocky Ridge Drive, Suite 380
Roseville, CA 95661
USA
Phone: 1.916.928.6400
Toll Free: 1.800.468.6439
Fax: 1.916.928.6404
Germany: +49-(0)89-208039 – 158
United Kingdom: +44 (0)1753 245 510
France: +33 (0)1•72•86•60•77