HOWTO: Connecting Perl to SQL Server

This HOWTO document provides examples for connecting a Perl script to a Microsoft SQL Server database. It is divided into two sections, and you will need to refer to the appropriate section depending on if you are running your Perl script on a Windows platform or a Unix platform.

1. Windows

If you are running your Perl script on a Windows platform, it is recommended that you use one of two packages that rely on DBI, the standard database interface module for Perl.

DBD::ODBC
DBD::ADO

I was able to install these packages without hassle using the ActiveState PPM tool. If you install either of these two packages, PPM will also install their prerequisite package DBI, if it is not already installed. Installing the DBD::ADO package will also install the Win32::OLE package.

1.1 Using DBD::ODBC

If you choose to use DBD::ODBC, the following sample code will explain how to connect to a SQL Server database.

```
use DBI;

# DBD::ODBC

my $dsn = 'DBI:ODBC:Driver={SQL Server}';
my $host = '10.0.0.1,1433';
my $database = 'my_database';
my $user = 'sa';
my $auth = ‘s3cr3t';

# Connect via DBD::ODBC by specifying the DSN dynamically.
my $dbh = DBI->connect("$dsn;Server=$host;Database=$database",
                       $user,
                       $auth,
                       { RaiseError => 1, AutoCommit => 1}
                      ) || die "Database connection not made: $DBI::errstr";

#Prepare a SQL statement
```

```
my $sql = "SELECT id, name, phone_number FROM employees ";
my $sth = $dbh->prepare( $sql );

#Execute the statement
$sth->execute();

my( $id, $name, $phone_number );

# Bind the results to the local variables
$sth->bind_columns( undef, \$id, \$name, \$phone_number );

#Retrieve values from the result set
while( $sth->fetch() ) {
  print "$id, $name, $phone_number\n";
}

#Close the connection
$sth->finish();
$dbh->disconnect();
```

You can also connect using a System DSN that you have set up in advance. To set up a System DSN you can access it at Control Panel > Administrative Tools > Data Sources. To use your System DSN to connect, you simply need to alter the connect string in the above example to look like this:

```
# Connect via DBD::ODBC using a System DSN
my $dbh = DBI->connect("dbi:ODBC:my_system_dsn",
                       $user,
                       $auth,
                       {
                         RaiseError => 1,
                         AutoCommit => 1
                       }
                      ) || die "Database connection not made: $DBI::errstr";
```

1.2 Using DBD::ADO

If you choose to use the DBD::ADO package, the following sample code will explain how to connect to a SQL Server database.

```
use DBI;

my $host = '10.0.0.1,1433';
my $database = 'my_database';
my $user = 'sa';
my $auth = ‘s3cr3t';

# DBD::ADO
$dsn  = "Provider=sqloledb;Trusted Connection=yes;";
$dsn .= "Server=$host;Database=$database";
my $dbh = DBI->connect("dbi:ADO:$dsn",
                       $user,
                       $auth,
                       { RaiseError => 1, AutoCommit => 1}
                      ) || die "Database connection not made: $DBI::errstr";

#Prepare a SQL statement
my $sql = "SELECT id, name, phone_number FROM employees ";
```

```
my $sth = $dbh->prepare( $sql );

#Execute the statement
$sth->execute();

my( $id, $name, $phone_number );

# Bind the results to the local variables
$sth->bind_columns( undef, \$id, \$name, \$phone_number );

#Retrieve values from the result set
while( $sth->fetch() ) {
  print "$id, $name, $phone_number\n";
}

#Close the connection
$sth->finish();
$dbh->disconnect();
```

2. Unix

If you are running your Perl script on a Unix platform, the DBD::Sybase package is required to connect to a SQL Server database.

2.1. Install SQL Server Library Support

The Sybase DBD package requires the FreeTDS driver.

The FreeTDS driver can be downloaded at the following location.

http://www.freetds.org/

Instructions for building the FreeTDS driver can be found at the following location.

http://www.freetds.org/userguide/config.htm

Because the driver is not being used with ODBC, the **-disable-odbc** switch can be used with the **configure** command while installing FreeTDS.

2.2 Configure the Data Source

Next, the freetds.conf file should be modified to include the SQL Server database information. A sample entry is shown below:

[SS_MY_DB]
host = 10.0.0.1 # or host name

port = 1433
tds version = 7.0

2.3 Install the Sybase DBD Module

For more information on installing this Perl module, refer to http://search.cpan.org/~mewp/DBD-Sybase/Sybase.pm.

In addition, the **SYBASE** environment variable should be set to the location of the FreeTDS installation.

export SYBASE=/usr/local/freetds

2.4 Sample Code using the Sybase DBI and a SQL Server DSN

```
#  load the DBI module
use DBI;
use DBD::Sybase;


my $database="my_database";
my $user="sa";
my $auth="s3cr3t";


BEGIN
{
    $ENV{SYBASE} = "/usr/local";
}

# Connect to the SQL Server Database
my $dbh = DBI->connect("dbi:Sybase:server=ss_my_db;database=$database",
                        $user,
                        $auth
                        {RaiseError => 1, AutoCommit => 1}
                      ) || die "Database connection not made: $DBI::errstr";

#Prepare a SQL statement
my $sql = "SELECT id, name, phone_number FROM employees";
my $sth = $dbh->prepare( $sql );

#Execute the statement
$sth->execute();

my( $id, $name, $phone_number );

# Bind the results to the local variables
$sth->bind_columns( undef, \$id, \$name, \$phone_number );

#Retrieve values from the result set
while( $sth->fetch() ) {
```

```
  print "$name, $title, $phone\n";
}

#Close the connection
$sth->finish();
undef $sth; # This fixes a segfault bug with certain versions of DBD::Sybase
$dbh->disconnect();
```

3 Source

http://www.microsoft.com/technet/itsolutions/cits/interopmigration/unix/oracleunixtosql/or2sql12.mspx