Subject: Re: MSSQL access from Linux
Posted by mirek on Mon, 19 Oct 2009 12:11:51 GMT
View Forum Message <> Reply to Message

chickenk wrote on Mon, 19 October 2009 05:34Hi,

I would like to know if some access to a MSSQL server from a linux box is possible using U++. I know that the library FreeTDS (http://www.freetds.org/) does it, so I wondered if Mirek got something working ?

Yes. It works. Current MSSQL package uses ODBC and works in Linux.

All you need to do is to configure the damn thing, which is quite tricky....

Anyway, here is a piece of code I use to connect (direct copy&paste from the app, so you will have to sort that out :):

```
#ifdef flagODBC
bool Login(ODBCSession& session)
#else
bool Login(OleDBSession& session)
#endif
{
 LoginDlg dlg;
 dlg.Icon(HollyImg::holly());
 LoadFromFile(dlg, ConfigFile("login.cfg"));
 String s = GetIniKey("DATABASE");
 if(s.GetCount())
  dlg.database <<= s;
#ifndef _DEBUG
 dlg.password <<= Null;
 dlg.password.Password();
#endif
 for(;;) {
  if(dlg.Run() != IDOK) {
   StoreToFile(dlg, ConfigFile("login.cfg"));
   return false;
  }
 #ifdef PLATFORM_WIN32
  String provider = Nvl(GetIniKey("CONNECT"), "SQL Server Native Client 10.0");
 #else
  String provider = Nvl(GetIniKey("CONNECT"), "MSSQL");
 #endif
 #ifdef flagODBC
  #ifdef PLATFORM_POSIX
   String dfn = "/usr/lib/libtdsodbc.so.*";
```

```
  SetIfExists(dfn, "/usr/local/lib/libtdsodbc.so.*");
  SetIfExists(dfn, GetHomeDirFile("libtdsodbc.so.*"));
  SaveFile(GetHomeDirFile(".odbc.ini"),
   String().Cat() << "[MSSQL]\n"
    << "Driver = " << dfn << '\n'
    << "Description = MSSQL\n"
    << "ServerName = MSSQL\n"
  );
  String c =
   "[MSSQL]\n"
   "tds version = 8.0\n"
   "client charset  = WINDOWS-1250\n"
  ;
  String host = ~dlg.database;
  String port;
  int q = host.ReverseFind(',');
  if(q >= 0)
   c << "host = " << host.Mid(0, q) << '\n'
     << "port = " << host.Mid(q + 1) << '\n';
  else
   c << "host = " << host << '\n';
  SaveFile(GetHomeDirFile(".freetds.conf"), c);
  String cs = "DSN=MSSQL;SERVER=MSSQL;";
 #else
  String cs = "Driver={SQL Server Native Client 10.0}";
  cs << ";Server=" << ~dlg.database << ';';
 #endif
 if(dlg.windows)
  cs << "Trusted_Connection=Yes;";
 else
  cs << "UID=" << ~dlg.username << ";PWD=" << ~dlg.password << ';';
 if(session.Connect(cs))
  break;
#else
 String propset;
 propset << "Provider=" << provider << ";Data Source=" << ~dlg.database
         << ";Trusted_Connection=Yes;";
 if(dlg.windows ? session.OpenProp(propset) :
    session.Open(~dlg.username, ~dlg.password, ~dlg.database, provider))
  break;
#endif
 Exclamation("[A4 Connection failed!]&[A1 " + DeQtfLf(session.GetLastError()));
}
StoreToFile(dlg, ConfigFile("login.cfg"));
return true;
}
```

The nasty trick used there is that login process creates some .ini file for FreeTDS

Mirek