Betrifft	Verwendung von Windows Systemlibraries als externe Prozeduren unter 9*i*
Autor	Jens-Uwe Petersen (jens.petersen@trivadis.com)
Art der Info	Technische Background Info (Dezember 2003)
Quellen	Oracle8 Externe Procedures aus PL/SQL (Urs Meier, Trivadis Publikation) Externe Prozeduren unter Windows mit dem Freeware-Compiler *MinGW* (Jens-Uwe Petersen, Trivadis Publikation)

Einleitung

Wer schon einmal versucht hat die Beispiele aus dem Artikel Oracle8 Externe Procedures aus PL/SQL auf einer 9*i*-Datenbank (9.0.1.x bis 9.2.0.x) zu installieren, musste wahrscheinlich irritiert feststellen, dass diese beim Aufruf mit einem Fehler abbrechen:

```
ORA-28576: lost RPC connection to external procedure agent
```

Zunächst sollten natürlich Fehler in der Konfiguration ausgeschlossen werden (u. a. dadurch, dass die von Oracle mitgelieferten Beispiele einwandfrei funktionieren). Falls der Fehler aber weiterhin hartnäckig besteht, handelt es sich mit ziemlicher Sicherheit um den nichtöffentlichen Bug 2379740 und der betrifft alle Versionen von Oracle 9*i* unter Windows NT, 2000 und XP.

Das Problem besteht darin, dass es unter Windows mehrere Aufrufkonventionen gibt:

Aufrufkonvention	Linker Direktive	Zugehöriger Calling Standard in PL/SQL	Bemerkung
C declaration	_cdecl	C	Standard bei C-Programmen
Standard call	_stdcall	PASCAL	Verwendung für Windows-API

Diese Konventionen legen fest in welcher Reihenfolge die Parameter auf dem Stack abgelegt werden und wer nach Beendigung des Funktionsaufrufs den Stack wieder aufräumt. Bei Verwendung von *Standard Call* räumt die aufgerufene Funktion am Ende den Stack auf, während bei *C declaration* die aufrufende Funktion das Aufräumen übernimmt und außerdem die Parameter in umgekehrter Reihenfolge auf dem Stack abgelegt werden.

Nun ist es so, dass externe Prozeduren unter Windows momentan nur funktionieren, wenn die DLL mit der Aufrufkonvention *_cdecl* erstellt wurde. Die Systemlibraries wurden von Microsoft aber mit der Aufrufkonvention *_stdcall* erstellt. Das sollte zwar eigentlich egal sein, denn schließlich kann man beim Deklarieren der Prozeduren in PL/SQL den CALLING STANDARD angeben und hier zwischen *C* und *PASCAL* wählen. Nur leider wurde die *PASCAL*-Aufrufkonvention laut interner Bugbeschreibung nicht vollständig implementiert und funktioniert darüber hinaus auch nicht.

Als Workaround wird von Oracle daher immer empfohlen, die DLL mit Konvention *_cdecl* nochmals neu zu erstellen. Soweit man im Besitz der Sourcen ist, ist das auch ein praktikabler Vorschlag, nur hat man diese eben bei den Windows-APIs oder sonstigen gekauften DLLs leider nicht zur Verfügung.

In diesen Fällen bleibt momentan nur, sich eine Wrapper-DLL zu schreiben, die von EXTPROC aufgerufen wird, die ursprüngliche DLL lädt und daraus die gewünschte Funktion ausführt. Wie das funktioniert, soll nun im Einzelnen beschrieben werden.

Erstellen der Wrapper-DLL

Für unser Beispiel schauen wir, dass wir die beiden Funktionen *GetComputerName* und *GetDriveType* aus der Windows-Library *kernel32.dll*, die im oben erwähnten Artikel verwendet wurden, wieder aus PL/SQL aufrufen können.

Das Projektverzeichnis für unser Beispiel soll *'C:\Programme\MinGW\Projects\tvd_beispiele'* sein, dort speichern wir den nachfolgenden Quelltext unter dem Namen *tvdbsp_kernel32.c*

Um Probleme mit Namenskonflikten bei den Funktionen zwischen den DLLs zu vermeiden, verwenden wir die gleichen Funktionsnamen und setzen noch ein *'tvd'* vorne dran.

```
#include <stdio.h>
#include <windows.h>
/* ------------------------------------------------------------------- */
// Source: tvdbsp_kernel32.c
/* ------------------------------------------------------------------- */
//Definition function prototype
typedef short (CALLBACK* GetComputerNameAType)(LPTSTR, LPDWORD);

__declspec(dllexport)
short __cdecl tvdGetComputerNameA ( char *pReturn1, long *pReturn1_ind )
{
   HINSTANCE               dllHandle;
   GetComputerNameAType    GetComputerNameAPtr;
   BOOL                    freeResult = FALSE;
   short                   retCode  = -1;

   // Hole Handle fuer die DLL.
   dllHandle = LoadLibrary("kernel32");

   // Wenn Handle gueltig, ermittle Adresse der Funktion.
   if (dllHandle != NULL) {
      GetComputerNameAPtr = (GetComputerNameAType)
                               GetProcAddress(dllHandle, "GetComputerNameA");

      // Wenn Adresse gueltig, rufe die Funktion auf.
      if (NULL != GetComputerNameAPtr) {
         retCode = (GetComputerNameAPtr) ((LPTSTR)pReturn1, pReturn1_ind);
      }

      // Aufraeumen
      freeResult = FreeLibrary(dllHandle);
   }
   return retCode;
}

/* ------------------------------------------------------------------- */
//Definition function prototype
typedef short (CALLBACK* GetDriveTypeAType)(LPTSTR);
```

```
__declspec(dllexport)
short __cdecl tvdGetDriveTypeA ( char *pReturn1 )
{
   HINSTANCE            dllHandle;
   GetDriveTypeAType    GetDriveTypeAPtr;
   BOOL                 fFreeResult = FALSE;
   short                retCode  = -1;

   // Hole Handle fuer die DLL.
   dllHandle = LoadLibrary("kernel32");

   // Wenn Handle gueltig, ermittle Adresse der Funktion.
   if (dllHandle != NULL) {
      GetDriveTypeAPtr = (GetDriveTypeAType)
                           GetProcAddress(dllHandle, "GetDriveTypeA");

      // Wenn Adresse gueltig, rufe die Funktion auf.
      if (NULL != GetDriveTypeAPtr) {
         retCode = (GetDriveTypeAPtr) ((LPTSTR)pReturn1);
      }

      // Aufraeumen
      fFreeResult = FreeLibrary(dllHandle);
   }
   return retCode;
}
/* ------------------------------------------------------------------ */
```

Zur Erzeugung der DLL wird entweder Microsofts Visual C++ oder wie in unserem Fall der MinGW Compiler verwendet (die Konfiguration ist in Externe Prozeduren unter Windows mit dem Freeware-Compiler *MinGW* beschrieben).

```
gcc -shared -o tvdbsp_kernel32.dll tvdbsp_kernel32.c
```

Um die DLL in PL/SQL referenzieren zu können, nun die zugehörige Library anlegen:

```
CREATE OR REPLACE LIBRARY lib_kernel32 AS
'C:\Programme\MinGW\Projects\tvd_beispiele\tvdbsp_kernel32.dll';
/
```

Anschließend die Funktionen in PL/SQL definieren:

```
CREATE OR REPLACE FUNCTION getcomputername ( lpbuffer IN OUT VARCHAR2,
                                             nsize    IN OUT BINARY_INTEGER
                                           ) RETURN BINARY_INTEGER
IS EXTERNAL
LIBRARY  lib_kernel32
NAME     "tvdGetComputerNameA"
LANGUAGE C
CALLING  STANDARD C;
/

CREATE OR REPLACE FUNCTION getdrivetype ( lpbuffer IN OUT VARCHAR2 )
RETURN BINARY_INTEGER
IS EXTERNAL
LIBRARY  lib_kernel32
NAME     "tvdGetDriveTypeA"
LANGUAGE C
CALLING  STANDARD C;
/
```

und dann kann die ordnungsgemäße Arbeitsweise der beiden Funktionen überprüft werden:

```
set serveroutput on;
DECLARE
   rwert BINARY_INTEGER;
   slen  BINARY_INTEGER := 255; -- must be LENGTH of sbuf
   sbuf  VARCHAR2(255)  := ' '; -- may not be NULL
BEGIN
   rwert := getcomputername(sbuf,slen);
   dbms_output.put_line(sbuf);
END;
/

DECLARE
rwert BINARY_INTEGER;
sbuf  VARCHAR2(255) := '&drivename'; -- may not be NULL, e.g. c:
BEGIN
   rwert := getdrivetype(sbuf);
   CASE rwert
      WHEN 2 THEN dbms_output.put_line(sbuf || ' is a removable drive');
      WHEN 3 THEN dbms_output.put_line(sbuf || ' is a fixed drive');
      WHEN 4 THEN dbms_output.put_line(sbuf || ' is a remote drive');
      WHEN 5 THEN dbms_output.put_line(sbuf || ' is a CD-ROM drive');
      WHEN 6 THEN dbms_output.put_line(sbuf || ' is a RAM-Disk drive');
      ELSE dbms_output.put_line('Don''t know this drive');
   END CASE;
END;
/
```

Zusammenfassung

Die Informationspolitik von Oracle in diesem Fall ist als ausgesprochen schlecht zu bezeichnen.
Anwender, die in MetaLink Hilfe zu diesem Thema suchten, bekamen stets als Antwort das sei ein reines Problem von Microsoft, Oracle könne hier nicht helfen. Der Bug ist bis heute non-public und es gibt auch keinerlei Hinweise ob und wann die *PASCAL*-Aufrufkonvention implementiert wird. Da der Bug aber mit Priorität 3 eingestuft wurde (minimal loss of service), kann das erfahrungsgemäß dauern.
Der gezeigte Workaround kann nur eine Notlösung sein, da er bei einer größeren Anzahl von benötigten Funktionen in einer DLL einen erheblichen Mehraufwand mit sich bringt.

Jens-Uwe Petersen

Trivadis GmbH
Jens-Uwe Petersen
Max-Lang-Strasse 56
70771 Leinfelden-Echterdingen

Mail: jens.petersen@trivadis.com
Tel: +49 711 903 63 230
Fax: +49 711 903 63 259
Internet: http://www.trivadis.com