

---

Subject: Re: Is it possible to call a stored procedure with an output parameter?

Posted by [jjacksonRIAB](#) on Fri, 17 Feb 2012 20:49:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

I tried to figure out how I could get this working in U++ but I can't get return values out of MSSQL.

So far under ODBC.cpp I made these changes:

```
struct Param {
    int  ctype;
    int  sqltype;
    String data;
    SQLENN li;
    int  direction;
};
```

Then I modified ODBCConnection::Execute and I'm still trying to get this to work:

```
bool ODBCConnection::Execute()
{
    bool isStoredProcedure = false;

    LLOG("Execute " << (void *)this << " " << (void *)session);
    if(session->hstmt == SQL_NULL_HANDLE)
        return false;
    if(IsCurrent())
        session->current = NULL;
    session->FlushConnections();
    last_insert_table.Clear();
    number.Clear();
    text.Clear();
    time.Clear();
    CParser p(statement);

    // parse for evidence of a stored procedure call
    if(p.Char('{'))
    {
        p.Spaces();

        if(p.Id("call") || p.Id("CALL")) {
            procedure_name = p.ReadId();
            isStoredProcedure = true;
            //Cout() << "Proc name: " << procedure_name << "\n";
        }
    }
```

```
SDWORD    cbValue5;
SDWORD    cbValue4;

SQLSMALLINT ParameterType = SQL_PARAM_INPUT;
```

```
if(!IsOk(SQLProcedureColumns (
    session->hstmt,
    NULL,
    0,
    NULL,
    0,
    (SQLCHAR *)~procedure_name,
    procedure_name.GetLength(),
    NULL,
    0
)))
{
    SQLFreeStmt(session->hstmt, SQL_CLOSE);
    return false;
}
```

```
char parameter_name [20];
```

```
if(!IsOk(SQLBindCol(
    session->hstmt,
    4, // Column 5 returns column name
    SQL_C_CHAR,
    parameter_name,
    sizeof(parameter_name),
    &cbValue4
)))
{
}
```

```
if(!IsOk(SQLBindCol(
    session->hstmt,
    5, // Column 5 returns whether parameter is input or output
    SQL_C_SHORT,
    &ParameterType,
    0,
    &cbValue5
)))
{
}
```

```
int i = 0;
while(SQLFetch(session->hstmt) == SQL_SUCCESS)
{
```

```

Param& p = param[i];

switch (ParameterType)
{
    case SQL_PARAM_INPUT:
    case SQL_PARAM_OUTPUT:
    case SQL_PARAM_INPUT_OUTPUT:
        p.direction = ParameterType;
        i++;
        break;

    default:
        break;
}
}

if(IsCurrent()) session->current = NULL;
session->FlushConnections();

number.Clear();
text.Clear();
time.Clear();
}

if((p.Id("insert") || p.Id("INSERT")) && (p.Id("into") || p.Id("INTO")) && p.IsId())
    last_insert_table = p.ReadId();
if(!IsOk(SQLPrepare(session->hstmt, (SQLCHAR *)~statement, statement.GetCount()))
    return false;
parse = false;
bparam = param;
param.Clear();
for(int i = 0; i < bparam.GetCount(); i++) {
    Param& p = bparam[i];
    SQLSMALLINT    DataType;
    SQLULEN        ParameterSize;
    SQLSMALLINT    DecimalDigits;
    SQLSMALLINT    Nullable;

    if(!IsOk(SQLDescribeParam(session->hstmt, i + 1, &DataType, &ParameterSize,
&DecimalDigits, &Nullable)))
        return false;

    Cout() << "Param direction: " << p.direction << "\n";

    if(!IsOk(SQLBindParameter(session->hstmt, i + 1, p.direction, p.ctype, DataType,
        ParameterSize, DecimalDigits, (SQLPOINTER)~p.data,
        p.data.GetLength(),
        &p.li))) {

```

```

    return false;
}

}
SQLSMALLINT ncol;

SQLExecute(session->hstmt);

if(isStoredProcedure) {
    while (SQLMoreResults(session->hstmt)!= SQL_NO_DATA );
}

if(!(*!isOk(SQLExecute(session->hstmt)) ||*! !isOk(SQLNumResultCols(session->hstmt, &ncol)))) {

    SQLFreeStmt(session->hstmt, SQL_CLOSE);
    return false;
}

session->current = this;
info.Clear();
binary.Clear();
for(int i = 1; i <= ncol; i++) {
    SQLCHAR    ColumnName[256];
    SQLSMALLINT NameLength;
    SQLSMALLINT DataType;
    SQLULEN    ColumnSize;
    SQLSMALLINT DecimalDigits;
    SQLSMALLINT Nullable;
    if(!isOk(SQLDescribeCol(session->hstmt, i, ColumnName, 255, &NameLength, &DataType,
        &ColumnSize, &DecimalDigits, &Nullable)))
        return false;
    binary.Add(false);
    SqlColumnInfo& f = info.Add();
    f.nullable = Nullable != SQL_NO_NULLS;
    f.binary = false;
    f.precision = DecimalDigits;
    f.scale = 0;
    f.width = ColumnSize;
    f.name = (char *)ColumnName;
    switch(DataType) {
    case SQL_DECIMAL:
    case SQL_NUMERIC:
    case SQL_SMALLINT:
    case SQL_INTEGER:
    case SQL_REAL:
    case SQL_FLOAT:
    case SQL_DOUBLE:

```

```

case SQL_BIT:
case SQL_TINYINT:
    f.type = DOUBLE_V;
    break;
case SQL_BIGINT:
    f.type = INT64_V;
    break;
case SQL_TYPE_DATE:
case SQL_TYPE_TIMESTAMP:
    f.type = TIME_V;
    break;
case SQL_BINARY:
case SQL_VARBINARY:
case SQL_LONGVARBINARY:
    f.type = STRING_V;
    f.binary = true;
    binary.Top() = true;
    break;
default:
    f.type = STRING_V;
    break;
}
}
SQLLEN rc;
SQLRowCount(session->hstmt, &rc);
rowsprocessed = rc;

return true;
}

```

I am calling `SQLProcedureColumns` to find out whether a given bound column is an input column, output column, or bidirectional. They say this is an expensive call so I'm considering wrapping it in a different method that will cache those parameters for later. These changes also do not bind return values (yet), just output parameters.

The problem I'm having is that even after calling `SQLMoreResults` (after `SQLExecute`), the documentation says those bound parameters should be modified, yet they aren't being changed - I can look through and confirm that all the direction types are correct.

Where did I go wrong? Am I doing this stupidly?