**Technical Note**

# Executing SQL Server Stored Procedures with Rep++

## Overview

A stored procedure is a subroutine (or procedure) that is physically stored within a database. It is modular and runs directly on the database engine, which is generally faster at processing database requests.

The exact implementation of stored procedures varies from one database system to another. Most major database vendors support them in some form. They are usually written in a proprietary database language like T-SQL for Microsoft SQL Server, PL/SQL for Oracle database or PL/PgSQL for PostgreSQL.

This article describes how to execute SQL Server stored procedures with Rep++ and how to get scalar or result set return values.

# Executing SQL Server Stored Procedures with Rep++

## Prerequisites

- Rep++ installed, along with the DEMOV2 system.
- SQL Server as database.
- Microsoft SQL Server Management Studio installed and nowledge of how to create stored procedures in that environment.

## Executing stored procedures that do not return any value

In this section, you will create a simple stored procedure that attempts to create a new client. Then, you will write the code that uses Rep++ to call your stored procedure. If the client already exists, you will display a message box containing the returned error message.

1. Open Microsoft SQL Server Management Studio.
2. In the Object Explorer, open the Databases node, then the DEMO node (or the one containing the DEMOV2 data), and create the CreateClient stored procedure as described below.

```
CREATE procedure CreateClient (
  @ClientFirstName    varchar(40),
  @ClientLastName     varchar(40)
)
as
  declare    @Code    varchar(16)

  if len(@ClientFirstName) > 4
    Set @Code = substring(@ClientFirstName, 1, 4)
  else
    Set @Code = @ClientFirstName

  if len(@ClientLastName) > 4
    Set @Code = upper(@Code + substring(@ClientLastName, 1, 4))
  else
    Set @Code = upper(@Code + @ClientLastName)

  insert CT_Client   (ClientCode,
                      ClientFirstName,
                      ClientLastName,
                      ClientType,
                      ClientSalesToDate,
                      CreationDate,
                      ModificationDate)
  values (@Code,
          @ClientFirstName,
          @ClientLastName,
          6,
          0,
          getdate(),
          getdate())
```

```
GO
```

3. Create a new Windows® Forms application project.
4. Drop a RepPPInfo component on the default form and set its connection information.
5. Add a button to the default form and handle its Click event to call the CreateClient stored procedure.

```
private void button1_Click(object sender, EventArgs e) {
  RepPP.Application app;
  RepPP.Connection  connection;
  RepPP.SqlCommand  sqlCmd;
  RepPP.ErrorCode   errCodeResult;
  string            strSqlCmd;
  string            strMessage;
  bool              bSuccess = false;

  using (app = RepPP.Application.CreateFromRes()) {
    connection = app.DataConnection;
    strSqlCmd  = @"execute CreateClient @ClientFirstName=:P1,
                                        @ClientLastName=:P2";
    sqlCmd = connection.SqlCommands.Open(strSqlCmd);
    sqlCmd.SetParameterValue("P1", "Santa", RepPP.FieldType.sdFieldString);
    sqlCmd.SetParameterValue("P2", "Claus", RepPP.FieldType.sdFieldString);
    try {
      errCodeResult = (RepPP.ErrorCode)sqlCmd.Execute();
      if (errCodeResult != RepPP.ErrorCode.sdNoErr) {
        strMessage  = "Cannot execute the stored procedure";
        strMessage += "\nRep++ Error Code: " + errCodeResult;
        strMessage += "\nDB Error Code:    " + connection.ErrorCode;
        strMessage += "\nDB Error Message: " + connection.ErrorMessage;
        MessageBox.Show(strMessage);
      } else {
        MessageBox.Show("The stored procedure executed successfully!");
        bSuccess = true;
      }
    } finally {
      if (bSuccess) {
        connection.Commit();
      } else {
        connection.Rollback();
      }
    }
  }
}
```

6. Build your project and run the application.

The first time you click the button, the client will be created successfully. The second time, however, you will get an error message stating that the client exists already!

# Executing stored procedures that return scalar output parameters

You will now write a simple stored procedure that creates a new address for a given client and returns the auto-generated ID of the newly added address. Then, you will write the code that uses Rep++ to call your stored procedure and retrieve the value of the address ID.

1. Create the CreateAddress stored procedure.

```
CREATE procedure CreateAddress (
  @ClientCode           varchar(16),
  @AddressLine1         varchar(80),
  @City                 varchar(40),
  @PostalCode           varchar(6),
  @AddressCode          int           OUT
)
as
  begin
    Insert CT_Address(ClientCodeFK, Address_Line1, City, PostalCode)
    values (@ClientCode, @AddressLine1, @City, @PostalCode)
    -- Get Autoincrement Value
    Set @AddressCode = SCOPE_IDENTITY()
  end
GO
```

2. Add a second button to the default form and handle its Click event to call the CreateAddress stored procedure.

```
private void button2_Click(object sender, EventArgs e) {
  RepPP.Application   app;
  RepPP.Connection    connection;
  RepPP.SqlCommand    sqlCmd;
  RepPP.ErrorCode     errCodeResult;
  string              strSqlCmd;
  string              strMessage;
  bool                bSuccess    = false;

  using (app = RepPP.Application.CreateFromRes()) {
    connection     = app.DataConnection;
    strSqlCmd      = @"execute CreateAddress @ClientCode=:P1,
                                             @AddressLine1=:P2,
                                             @City=:P3,
                                             @PostalCode=:P4,
                                             @AddressCode=:P5 OUTPUT";
    sqlCmd          = connection.SqlCommands.Open(strSqlCmd);
    sqlCmd.SetParameterValue("P1","SANTCLAU",      RepPP.FieldType.sdFieldString);
    sqlCmd.SetParameterValue("P2","Christmas St.", RepPP.FieldType.sdFieldString);
    sqlCmd.SetParameterValue("P3","North Pole",    RepPP.FieldType.sdFieldString);
    sqlCmd.SetParameterValue("P4","H0H0H0",        RepPP.FieldType.sdFieldString);
    sqlCmd.SetParameterValue("P5",int.MaxValue.ToString(),
RepPP.FieldType.sdFieldNumeric);
    try {
      errCodeResult = (RepPP.ErrorCode)sqlCmd.Execute();
      if (errCodeResult != RepPP.ErrorCode.sdNoErr) {
        strMessage  = "Cannot execute the stored procedure";
        strMessage += "\nRep++ Error Code: " + errCodeResult;
        strMessage += "\nDB Error Code:    " + connection.ErrorCode;
        strMessage += "\nDB Error Message: " + connection.ErrorMessage;
        MessageBox.Show(strMessage);
      } else {
        strMessage  = "The stored procedure exected successfully.";
        strMessage += "\nNew address code: " + sqlCmd.GetParameterValue("P5");
        MessageBox.Show(strMessage);
        bSuccess    = true;
      }
    } finally {
      if (bSuccess) {
        connection.Commit();
      } else {
        connection.Rollback();
      }
    }
```

```
  }
}
```

In order to get the value of a stored procedure's scalar output parameter, you must allocate a buffer that is sufficient for its type. For instance:

- If the return value is an integer, your buffer must be able to hold the string representation of the largest integer (as in our code example).
- If the return value is a string (e.g. varchar(40)), your buffer must be able to hold the maximum number of characters allowed (i.e. 40).

Use the SqlCommand.SetParameterValue method to allocate a sufficient buffer for a scalar output parameter.

3. Build your project and run the application.

Every time you execute the CreateAddress stored procedure, an address is created and the auto-generated ID of the address is displayed.

# Executing stored procedures that return result sets

In this section, you will create a simple stored procedure that returns the client codes of all the clients. Then, you will write the code that uses Rep++ to call your stored procedure and retrieve the result set.

1. Create the GetClients stored procedure.

```
CREATE PROCEDURE GetClients
AS
BEGIN
  SELECT CLIENTCODE
  FROM CT_CLIENT;
END
GO
```

2. Add a third button to the default form and handle its Click event to call the GetClients stored procedure and fetch the client codes as indicated.

```
private void button3_Click(object sender, EventArgs e) {
  RepPP.Application  app;
  RepPP.SqlCursor    sqlCur = null;
  RepPP.ErrorCode    eErr;
  string             strMessage;

  using (app = RepPP.Application.CreateFromRes()) {
    eErr = (RepPP.ErrorCode)app.DataConnection.Execute("execute GetClients", out sqlCur);
      if (eErr == RepPP.ErrorCode.sdNoErr) {
        strMessage = "Clients:";
        while (sqlCur.Fetch() == (int)RepPP.ErrorCode.sdNoErr) {
          strMessage += "\n" + sqlCur.GetColumnValue("CLIENTCODE");
        }
        sqlCur.Close();
        MessageBox.Show(strMessage);
      }
  }
}
```

3. Build your project and run the application.

Every time you execute the GetClients stored procedure, a message box containing the list of client codes is displayed.