

## Tech Note 7

### Automatically Store Test Results in a Database

Terravic Research, Inc.

#### Introduction

ADODB and ODBC databases may be used to store test data of a fully automatic test procedure generated by Visual OPCTest Client. What will be accomplished includes: executing test cases of interest, determining the status of each executed test case, collecting all the failed test case data, storing information in a database, and finally notifying a developer by email of the failed results. Collecting test case data may be accomplished in several ways: external text file, csv file, xml file, or ADODB database connection (any other automation objects may be used for data storage as well). This Tech Note shows how to store test data in a database and how to send an email notification to a developer.

#### Platform Requirements

Visual OPCTest® Client 8.0.0000 or later.

Examples used in this Tech Note are based on the OPC Foundation sample servers. In order to successfully follow all the procedures outlined in this Tech Note, please download the OPC Foundation's Data Access sample server and the latest Visual OPCTest Client application. If the OPC Foundation sample server is not available, please substitute with another Data Access server and its tag definition. The substitution of server and tag definition in the sample code should be self-explanatory. Additionally, a database supported by the ADODB object must also be available to the Client, such as Microsoft Access, SQL Server, or Oracle.

#### Basic Steps

To setup a script that will automatically perform testing, data collection & storage, and email notification, we need to perform these five steps:

- (1) Create Database Table to Store Test Case Data
- (2) Setup ODBC Connection to the Database
- (3) Connect to the OPC Data Access Server
- (4) Run Validate Script Wizard
- (5) Modify the Wizard generated script based on our requirements

#### Step 1 - Create Database Table to Store Test Case Data

This example is based on Microsoft Access 2000 database. Information presented here may be used in setting up other databases, such as, SQL Server or Oracle. Any database filenames, location and table names may be changed at anytime. Please note that the sample script follows the naming conventions presented in this Tech Note.

Open Access, create blank database, and store it as *TestCaseData.mdb* in *C:\MyTests* directory. Double click on *Create table in Design view* selection.

Create following fields in the table:

<u>Field Name</u>	<u>Data Type</u>
TestCaseID	Text (Field Size 15, Allow Zero Length=No)

Status	Text (Field Size 15, Allow Zero Length=Yes)
ServerName	Text (Field Size 50, Allow Zero Length=Yes)
FunctionName	Text (Field Size 50, Allow Zero Length=Yes)
FunctionHR	Text (Field Size 25, Allow Zero Length=Yes)
OPCVersions	Text (Field Size 50, Allow Zero Length=Yes)
ParamIN	Text (Field Size 255, Allow Zero Length=Yes)
ParamOUT	Text (Field Size 255, Allow Zero Length=Yes)
ParamOUTLOOP	Text (Field Size 255, Allow Zero Length=Yes)
ParamCALLBACK	Text (Field Size 255, Allow Zero Length=Yes)
ExecTime	Text (Field Size 25, Allow Zero Length=Yes)
ExecDuration	Text (Field Size 10, Allow Zero Length=Yes)
Comment	Text (Field Size 255, Allow Zero Length=Yes)

Set *TestCaseID* field as the primary key (*Edit | Primary Key*).

Save table as `TestData_IOPCCCommon`.

Close the database.

## Step 2 - Setup ODBC Connection to the Database

Example of setting the ODBC connection to our database is based on Windows 2000 platform.

*Start, Settings, Control Panel, Administrative Tools*, [double-click on] *Data Sources (ODBC)*.  
*ODBC Data Source Administrator* dialog opens.

Select the *System DSN* tab.  
Click on the *Add...* button.  
Select *Microsoft Access Driver (.mdb)*.  
Click on the *Finish* button.

Data Source Name: *OPCTestData*  
Click on the *Select...* button, and point to *C:\MyTests\TestCaseData.mdb*; click the *OK* button.  
Click the *OK* button again.

Click on *OK* button to close *ODBC Data Source Administrator* dialog.

At this point we have created a database with one table to hold out test data. We have also created a system-wide ODBC connection to our database using the *ODBC Data Source Administrator* dialog.

## Step 3 - Connect to the OPC Data Access Server

Run the Client and select *Server | Connect to Server* menu item. The *OPC Server List* dialog will appear. For simplicity, select "OPC Data Access Servers Version 3.0" under LOCAL/ComputerName branch, then select "OPCSample.OpcDaServer.1" server. Click on the *Connect Server* button, then on the *Close* button. Of course, you may select a different server on either local or remote computer. Once we successfully connect to this server, there should be a server listed in the Workspace View with an entry such as "DA1\\LOCAL\\OPCSample.OpcDaServer.1" (this entry may be different based on your server location and name).

## Step 4 – Run Validate Script Wizard

We will use the Validate Script Wizard to get started with the base script. Once the Wizard completes, we will have a script code that connects the server, creates all the necessary groups, adds items to each group, sets up the test case tree, and disables/enables interfaces of interest.

Right-click on “DA1\\LOCAL\OPCSample.OpcDaServer.1” in the Workspace View, and select *Validate Script Wizard* from the popup menu.

The “Test Type Selection” wizard screen appears; keep default settings and click on the *Next* button.

The “Interface Test Selection” screen appears next. As we are executing specific test cases from a script, these settings are irrelevant to us. These options are useful when performing tests manually, therefore, accept the default selections and click on the *Next* button.

The “Test Group Selection” wizard screen appears next. This allows us to specify the different groups that the script will create for us, such as private or public, active or inactive, connected or not connected. All these groups are needed for Client to obtain predefined test case data for the standard test cases. Since this OPC Server does not support Advise Sink, deselect all options with Advise Sink (4, 5, and 6 from top), click on the *Next* button.

The “Test Item Selection” wizard screen appears. All the selected tags in this screen will be added to the groups defined in the previous wizard screen. Click on the *Browse* button to see the server’s address space. Double click on the following tags to select them:

- "Analog Types/Double"
- "Analog Types/Int"
- "Enumerated Types/Fellowship"
- "Limited Access/Read Only 1"
- "Limited Access/Write Only 1"
- "Simple Types/Float"
- "Simple Types/Int"

Click the *OK* button, and then click the *Next* button.

The “Finish – Generate Script” wizard screen appears last. Accept the defaults and click on the *Finish* button. Our script appears in the Script view.

Select from the main menu *File | Save As Script*, type “DatabaseStoreTest” as the file name, and click the *Save* button. Let’s test what our script creates. Select *Script | Run Script* from the main menu. Server is connected, all groups created and items are added, and finally, the test case tree is created as well. We are ready to modify the script.

## Step 5 – Modify Script

### *Simple Test Requirements for Storing Test Data in a Database*

Let’s setup a common ground of what we want to accomplish. First, we will be testing a Data Access server and we want to concentrate on an IOPCCommon interface. As Visual OPCTest Client has over 380 test cases just for testing IOPCCommon interface, we will base our tests on these standard test cases. Since we are testing a specific interface, we are only interested in collecting data of all the failed/warning test cases (when performing full integration test, all the test data may be collected, regardless of test case status). While executing all the test cases of interest, test case data of any test status that is set to FAIL or WARNING will be placed in the database. After all test cases execute, an email notification will be send to a developer in charge of IOPCCommon interface.

### ***Execute Test Case and Store Data in the Database***

First we need to write a subroutine that will execute a given test case and collect its data when status is set to either fail or warning. This will allow us to encapsulate test case execution into a single subroutine so we can use it on different test cases. This subroutine accepts OPC Data Access server handle (Client scripting handle), database connection object, and a test case number in string format, such as "2.1.1.2". Comments are self-explanatory and are preceded by the REM keyword. If the test case status after execution is anything but FAIL or WARNING, then the subroutine simply exits. Otherwise, data is collected for the FAIL/WARNING test case and it is stored in the database. Additionally, counters are updated to hold the total number of failed and warning test case results.

```
Public Sub ExecTestCase(ServerID, Conn, TCN)

    REM just a flag to determine if test case passed or not
    TestCaseNotBad = TRUE

    REM execute the given test case
    OPC.ExecuteTestCase ServerID, TCN

    REM get the test case status
    Status = OPC.GetTestCaseStatus(ServerID, TCN)

    REM check if it failed/warning or ok
    REM keep track of fail/warning total count
    If StrComp( Status, "FAIL", vbTextCompare ) = 0 Then
        CountFail = CountFail + 1
        TestCaseNotBad = FALSE
    End If
    If StrComp( Status, "WARNING", vbTextCompare ) = 0 Then
        CountWarning = CountWarning + 1
        TestCaseNotBad = FALSE
    End If

    If TestCaseNotBad = TRUE Then
        REM test case status is not FAIL or WARNING
        REM so there is nothing for us to place in the
        database

        IsDisabled = OPC.IsTestCaseDisabled(ServerID, TCN)
        If IsDisabled Then
            OPC.ScriptOutputEntry TCN & " test case is
            disabled"
        End If

        Exit Sub
    End If

    REM collect all the data for this FAIL/WARNING test case
    ServerName = OPC.GetTestCaseServerName(ServerID, TCN)
    FunctionName = OPC.GetTestCaseFunctionName(ServerID, TCN)
    FunctionHR = OPC.GetTestCaseFunctionHRESULT(ServerID, TCN)
    OPCVersions = OPC.GetTestCaseOPCVersions(ServerID, TCN)
    ParamIN = OPC.GetTestCaseParamIN(ServerID, TCN)
```

```

ParamOUT = OPC.GetTestCaseParamOUT(ServerID, TCN)
ParamOUTLOOP = OPC.GetTestCaseParamOUTLOOP(ServerID, TCN)
ParamCALLBACK = OPC.GetTestCaseParamCALLBACK(ServerID, TCN)
ExecTime = OPC.GetTestCaseExecTime(ServerID, TCN)
ExecDuration = OPC.GetTestCaseExecDuration(ServerID, TCN)
Comment = OPC.GetTestCaseComment(ServerID, TCN)

REM create SQL statement to enter test case data into the
database
REM Note, if entry for the given test case already exists
in the database, it must be UPDATE-ed instead of INSERT-ed
strSQL_1 = "INSERT INTO TestData_IOPCCCommon "
strSQL_2 = "(TestCaseID, Status, ServerName, FunctionName,
FunctionHR, OPCVersions, ParamIN, ParamOUT, ParamOUTLOOP,
ParamCALLBACK, ExecTime, ExecDuration, Comment) "
strSQL_3 = "VALUES ("
strSQL_4 = "'" & TCN & "', '" & Status & "', '" & ServerName
& "', '" & FunctionName & "', '" & FunctionHR & "', '" &
OPCVersions
strSQL_5 = "', '" & ParamIN & "', '" & ParamOUT & "', '" &
ParamOUTLOOP & "', '" & ParamCALLBACK
strSQL_6 = "', '" & ExecTime & "', '" & ExecDuration & "', '"
& Comment & "')"
strSQL = strSQL_1 & strSQL_2 & strSQL_3 & strSQL_4 &
strSQL_5 & strSQL_6

REM place the data in the database
OPC.StatusPaneMsg "Storing Test Case " & TCN & " in
Database"
Conn.Execute strSQL

```

End Sub

### ***Send Email Notification Function***

After all the test cases are executed and database has been updated, a notification may be emailed to a developer in charge of IOPCCCommon development. The email system must be installed and available on the test machine, as a default source email address will be used (From). The destination email address is passed in as a parameter *EmailAddress*. This function creates the email system object and sets all the email parameters, such as destination address and message body. Once the email is send, the email system object is released from memory.

```

Sub NotifyDeveloperByEmail(EmailAddress, DatabaseName, TableName)

OPC.StatusPaneMsg "Generating email to " & EmailAddress

Set golApp = CreateObject("Outlook.Application")
Set objNewMail = golApp.CreateItem(olMailItem)
objNewMail.Recipients.Add EmailAddress
objNewMail.Subject = "Database Test Results Notification"

objNewMail.Body = "Test Results stored in a database " &
DatabaseName & " (ODBC Connection). Table name is " &

```

```

        TableName & ". Total FAIL test cases = " & CountFail & ".
        Total WARNING test cases = " & CountWarning

objNewMail.Send

Set objNewMail = Nothing
Set golApp = Nothing

End Sub

```

### ***Executing Test Cases of Interest***

Before test case execution may begin, we need to create the database connection via our system-wide ODBC definition:

```

Set Conn = CreateObject("ADODB.Connection")
Conn.Open "OPCTestData","",""

```

Set the counter variables to 0 before test case execution function calls. These values will be inserted into the email notification:

```

CountFail = 0
CountWarning = 0

```

After the supporting subroutines are coded, execution of the test cases may begin. All the test cases are executed from within the `Main` subroutine. Simply issuing the following call to run a test case will do the job:

```

TCN = "2.1.1.2"
call ExecTestCase( ServerID, Conn, TCN )

```

Here, we are defining a test case number as string that we wish to execute. Calling our local subroutine will actually execute the test case, and determine if test case status is set to FAIL or WARNING. If it is, then the test case data will be placed in the database and the next test case may be executed.

Once all the test cases of interest are executed, notifying a developer by email is achieved with the following four lines of code:

```

EmailAddress = "Developer@TheCompanyXYZ.com"
DatabaseName = "OPCTestData"
TableName = "TestData_IOPCCommon"

NotifyDeveloperByEmail(EmailAddress,DatabaseName,TableName)

```

Just before our script completes, we should release the database connection object and remove it from memory.

```

Conn.Close
Set Conn = Nothing

```

Please note, only test cases with status set to FAIL or WARNING are placed in the database (as defined by our `ExecTestCase` subroutine). Changing this subroutine to store all the test case data may be accomplished by removing the initial `If` statement.

## Conclusion

By extending the script generated by the Validate Script Wizard, it is possible to truly customize the test case execution and report generation. Executing individual test cases may be replaced by executing individual functions or even entire interfaces with just one line of code. However, executing individual test cases, gives us the most flexibility in collecting test case data. Besides storing data in external database, we can place collected test case data in a text files or on the web.

## Sample Code File

Complete sample code file: `MyDatabaseStoreTest.otx`  
Sample code file may be found upon installation of the Visual OPCTest Client software.

## Contact

Contact Terravic with any questions regarding Visual OPCTest Client. Please let us know your concerns, desired improvements, or current issues with our software. We pride ourselves in making changes quickly based on your input. Please contact us at `support @ terravic.com`.

[www.opctest.com](http://www.opctest.com)  
[www.terravic.com](http://www.terravic.com)

