Let SAS Generate XML Code for ACCESS Audit Trail Data Macro

Sijian Zhang, Washington DC VA Medical Center

ABSTRACT

Microsoft ACCESS database is often used for data entry and data storage in small and medium sized projects. As a component of information security, the data audit trail application is usually required to be built in. Since ACCESS 2010, the data macro at the table level is available, which is very helpful for this purpose. However, the data macro setup can be very tedious when the number of variables is large. The alternative way is to export a small ACCESS macro as an XML file template, expand the template to all audit fields, and then import the expanded XML code back into the ACCESS database. This paper will give a brief introduction of audit trail in ACCESS and show how the XML code is generated with SAS macro facility.

INTRODUCTION

The audit trail file is a data operation record file. Its contents usually include, who changed, what field ("variable" and "field" are interchangeably used in this paper), in which table, with which ID (unique identifier), at what time. By viewing such an audit trail file, you will know the whole data entry and edit history of the variables under auditing. The following is a screenshot of an ACCESS audit trail table in the example project, see Figure 1.

tblAuditTrail_ScreenIC

ChangeID	TableName	FieldName	RecordID	OldValue	NewValue	ChangeDate	ChangeBy
17	tblSurveyScreenIC	S1	20709		1	8/25/2014 11:21:24 AM	VHAPTHRiceK
18	tblSurveyScreenIC	SIC_Status	20709		In Process	8/25/2014 11:21:24 AM	VHAPTHRiceK
19	tblSurveyScreenIC	S2	20709		0	8/25/2014 11:21:42 AM	VHAPTHRiceK
20	tblSurveyScreenIC	S3	20709		1	8/25/2014 11:24:34 AM	VHAPTHRiceK
21	tblSurveyScreenIC	S4	20709		1	8/25/2014 11:24:38 AM	VHAPTHRiceK
22	tblSurveyScreenIC	S5	20709		1	8/25/2014 11:24:44 AM	VHAPTHRiceK
23	tblSurveyScreenIC	S6	20709		1	8/25/2014 11:24:58 AM	VHAPTHRiceK

Figure 1: ACCESS example audit trail table

Let's look at the record with ChangeID=20 in the table above. The person with login name "VHAPTHRiceK" entered the option "1" (the value was changed to NewValue "1" from empty OldValue.) to the question "S3" for the subject with RecordID=20709 in the table "tblSurveyScreenIC" at "8/25/2014 11:24:23 AM". In other words, with this information, we can replay the process of data entering and editing. Besides its importance of information security, it is also very helpful to retrieve the old value if one accidentally deleted or changed a correct value.

This table is populated automatically during data entering and editing process, in which the data entry people may know nothing about it. In this example project, the audit trial information is saved in another linked ACCESS database which the data entry people cannot access.

Before ACCESS 2010, the audit trial process is done at form level. Its obvious issue is that it cannot track anything if the one bypasses the form, goes directly to the table and makes some changes. Since ACCESS 2010, the table level data macro is available, and is a very useful tool to handle data auditing tasks. Once it is set up correctly, the audit trail operation will catch any changes to the variables that you want to monitor, no matter in what way their values are changed. The macro is easy to set up through the following steps.

1. Create a table that logs the trail information, such as the table “tblAuditTrail_ScreenIC” in Figure 1. You can create one table for the whole database, or corresponding tables related to the target tables with the fields that you want to monitor.
2. Open the target table, and click the “After Update” option in the Table tab of Table Tools on the Ribbon, see Figure 2.

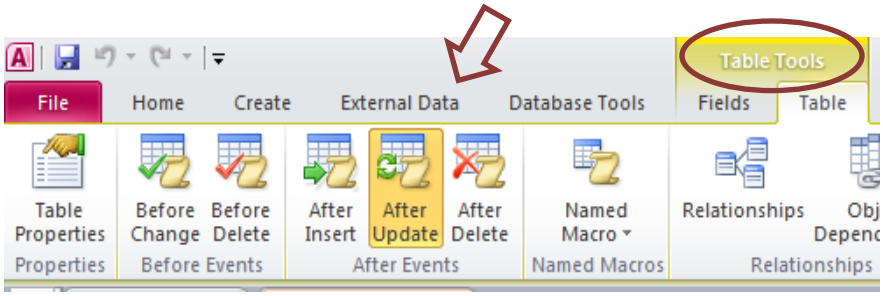

Figure 2: Options in Table Tools of ACCESS table view ribbon

3. After Step 2, you will see the data macro setup window, see Figure 3. Then, in the IF statement you create a record in the audit trail table for each value change of the monitored fields. Within record creation section, there are several SetField actions, which assign value change information to the fields in audit trail table.

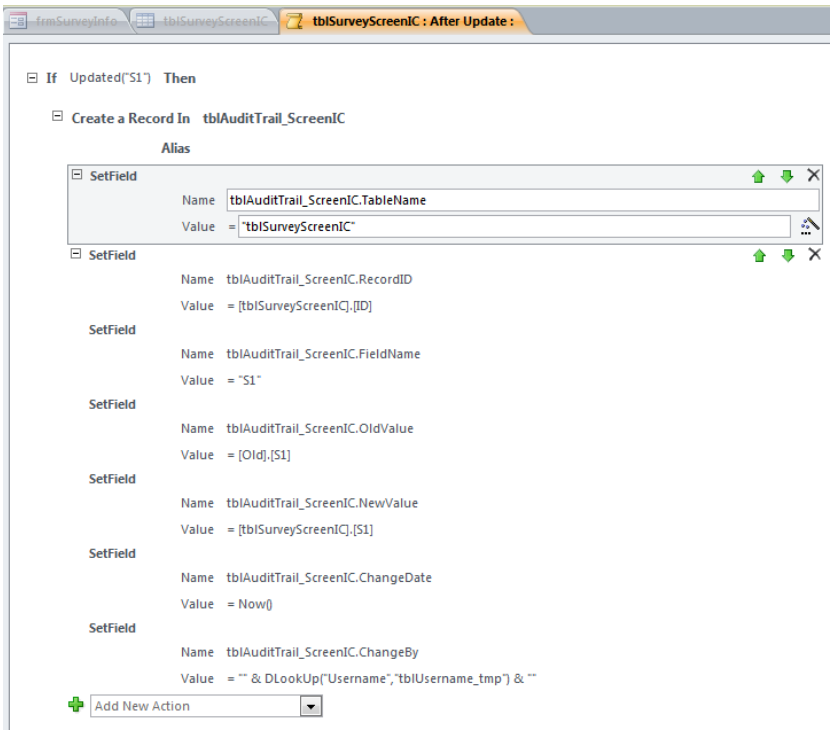

Figure 3: Data macro setup window

If you are going to monitor only a couple of fields, it's okay to repeat the steps above to set up the data macro. However, it will be a very tedious task if the number of fields is large. As a SAS programmer, you will naturally ask, "Can we use SAS macro to handle the repetitions?" The answer is "Yes, we can."

PROGRAMS

In order to take the advantage of SAS macro, we need first to get the source code for the audit application. Fortunately, it can be easily done using the following VBA code:

- Export the data macro to XML file:
  SaveAsText acTableDataMacro, "tblSurveyScreenIC", "C:\tmp\DataMacro_template.xml"

  This command exports the data macro for table "tblSurveyScreenIC" to an XML file "DataMacro_template.xml" saved in "C:\tmp\".

- Import the XML file to data macro:
  LoadFromText acTableDataMacro, "tblSurveyScreenIC", "C:\tmp\DataMacro_full.xml"

  This command imports "C:\tmp\DataMacro_full.xml" into the current database as a data macro for table "tblSurveyScreenIC".

The data macro is attached to a specific table, so the macro has no naming issue. But you need to name the XML file.

In the example project, I use the ACCESS data macro setup interface to set up one field, and then export it to an XML file as the template, “DataMacro_template.xml". According to the template, I write the macro, and then apply it to all the variables to generate the expanded XML file, “DataMacro_full.xml”, which is imported back to the database table, "tblSurveyScreenIC". If there is an existing data macro, it will be overwritten by the new one.

Now, let’s have a look at the data macro source code, “DataMacro_template.xml".

```
<?xml version="1.0" encoding="UTF-16" standalone="no" ?>
<DataMacros xmlns="http://schemas.microsoft.com/office/accessservices/2009/11/application">
<DataMacro Event="AfterUpdate">
<Statements>
<ConditionalBlock>
<If>
<Condition>Updated("S1")</Condition>
<Statements>
<CreateRecord>
<Data><Reference>tblAuditTrail_ScreenIC</Reference></Data>
<Statements>
<Action Name="SetField">
<Argument Name="Field">tblAuditTrail_ScreenIC.TableName</Argument>
<Argument Name="Value">"tblSurveyScreenIC"</Argument>
</Action>
<Action Name="SetField">
<Argument Name="Field">tblAuditTrail_ScreenIC.RecordID</Argument>
<Argument Name="Value">[tblSurveyScreenIC].[ID]</Argument>
</Action>
<Action Name="SetField">
<Argument Name="Field">tblAuditTrail_ScreenIC.FieldName</Argument>
<Argument Name="Value">"S1"</Argument>
</Action>
<Action Name="SetField">
<Argument Name="Field">tblAuditTrail_ScreenIC.OldValue</Argument>
<Argument Name="Value">[Old].[S1]</Argument>
</Action>
<Action Name="SetField">
<Argument Name="Field">tblAuditTrail_ScreenIC.NewValue</Argument>
<Argument Name="Value">[tblSurveyScreenIC].[S1]</Argument>
</Action>
<Action Name="SetField">                                                    ❶
<Argument Name="Field">tblAuditTrail_ScreenIC.ChangeDate</Argument>
<Argument Name="Value">Now()</Argument>
</Action>
<Action Name="SetField">
<Argument Name="Field">tblAuditTrail_ScreenIC.ChangeBy</Argument>
<Argument Name="Value">"" &amp;amp; DLookUp("Username","tblUsername_tmp") &amp;amp; ""</Argument>
</Action>
```

```
</Statements>
</CreateRecord>
</Statements>
</If>
</ConditionalBlock>
</Statements>
</DataMacro>
</DataMacros>
```

After going through the XML template, we can see that the tag <ConditionalBlock> is for the field monitored, which is easier to observe if the number of fields monitored in the template is two or more; and the tag <Action Name="SetField">, nested in <ConditionalBlock>, is to assign the field change related information to the fields in audit trail table. The following two macros are for these two tags respectively:

1. * Action on field in audit trail table *;

```
%macro fieldAction(fieldName,fieldValue);
put "<Action Name=""SetField"">";
put "<Argument Name=""Field"">&trailTable..&FieldName</Argument>";
put "<Argument Name=""Value"">&fieldValue</Argument>";
put "</Action>";
%mend fieldAction;
```

   This macro will generate the code that takes care of the repetitions of assigning value change imformation to each field in audit trail table, "tblAuditTrail_ScreenIC". For example, "%*fieldAction*(ChangeDate, Now())" will generate the code, see the framed portion ❶ in the above XML template, that assigns the current date and time value to the field "ChangeDate" in audit trail table. This macro is nested in the following macro *%block*( ).

2. * Conditional block for each field monitored *;

```
%macro block(var);
put "<ConditionalBlock>";
put "<If>";
put "<Condition>Updated(""&var"")</Condition>";
put "<Statements>";
put "<CreateRecord>";
put "<Data><Reference>&trailTable.</Reference></Data>";
put "<Statements>";
    * Table name *;
    %fieldAction(TableName,""&dataTable"")
    * Record ID *;
    %fieldAction(RecordID,[&dataTable].[ID])
    * Variable name *;
    %fieldAction(FieldName,""&var"")
    * Variable old value *;
```

```
    %fieldAction(OldValue,[Old].[&var])
    * Variable new value *;
    %fieldAction(NewValue,[&dataTable].[&var])
    * Change time *;
    %fieldAction(ChangeDate,Now())
    * Changed by log-in person *;
    %fieldAction(ChangeBy,%NRSTR(""""" &amp;amp; DLookUp
      (""Username"",""tblUsername_tmp"") &amp;amp; """""))
put "</Statements>";
put "</CreateRecord>";
put "</Statements>";
put "</If>";
put "</ConditionalBlock>";
%mend block;
```

This macro will generate the code that logs all audit information for one change of one monitored field. It will tackle the repeating work of all monitored fields one by one. The comments embedded in the code above tell what information is saved into the audit trail table. For example, "*%block*(S1)" will generate all the XML code between <ConditionalBlock> start and end tags shown in "DataMacro_template.xml" above. If this block of code is triggered in the table macro when any change happens to field "S1", all the relevant audit information will be recorded in the audit trail table.

Now, let's apply the macro *%block*( ) in a SAS Data step to generate the expected XML file. The parameter for this macro is the field name that we want to monitor. The table with the fields monitored is specified in the macro variable "&dataTable", while the audit trail table is specified in the macro variable "&trailTable". In the following SAS code, there are three sections in the DATA step for XML file generation, top: the heading, directly copied from the XML template file; middle: *%block*( ) macros for all fields to be monitored; and bottom: end tags corresponding to those in the heading section. The FILE statement ❷ is used to save the resulting XML file in a specified location.

```
%let dataTable=tblSurveyScreenIC;
%let trailTable=tblAuditTrail_ScreenIC;

data _null_;
    file "C:\tmp\DataMacro_full.xml";                ❷
    put '<?xml version="1.0" encoding="UTF-16" standalone="no" ?>';
    put '<DataMacros xmlns="http://schemas.microsoft.com/office/accessservices/2009/11/application">';
    put '<DataMacro Event="AfterUpdate">';
    put '<Statements>';
    %block(S1)
    %block(S2)
```

```
    %block(S3)
    .
    .
    .
    put '</Statements>';
    put '</DataMacro>';
    put '</DataMacros>';
run;
```

Once the expanded XML file, “DataMacro_full.xml”, is generated, it will be imported into the ACCESS database. After successful testing, this data macro will operated automatically in the background when the database is in use.

COMMENTS

- The XML file generated in SAS might miss something in the first several runs. So, you need to check by carefully comparing it against the XML template file before importing it into the database. If some bugs are not caught and imported, you will see warning signs or error messages in the ACCESS database testing.
- In the example of this paper, the audit trail covers only selected fields. If you want to cover all fields of a table, you can first use PROC SQL INTO statement to assign all field names into a SAS macro variable from DICTIONARY.COLUMNS for that table; then use a macro loop to run %block( ) through each of the field names. In this case, the middle section of the XML generation in SAS Data step will be a macro %do loop instead of individual %block( ) macros.
- If you want to add another layer of security to the audit information, as we do in this example project, you can create another ACCESS database in another server folder with limited access permission. Then you create a link to the external original audit trail table. In this way, you can periodically back up the audit trail data and empty the original audit trail table. Besides a better protection of the audit trail data, this way has another benefit that is to prevent the audit trail table size from growing too large in accumulation of the records over time.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Sijian Zhang, MS, MBA
Health Science Specialist
Chief of Staff Office
Washington DC VA Medical Center
50 Irving St. NW.
Washington, DC 20422
sijian.zhang@va.gov