

# Tracking Application Security Changes in EnterpriseOne

---

By Jeff Stevenson

---

THIS WHITE PAPER IS FOR INFORMATIONAL PURPOSES ONLY, AND MAY CONTAIN TYPOGRAPHICAL ERRORS AND TECHNICAL INACCURACIES. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND.

©Copyright 2006 Jeff Stevenson. All rights reserved. Reproduction in any manner whatsoever without the express written permission of Jeff Stevenson is strictly forbidden. For more information, see <http://www.thestevensons.org>.

Information in this document is subject to change without notice.

# Contents

---

<b>Executive Summary</b> .....	<b>3</b>
<b>Abstract</b> .....	<b>4</b>
<b>Problem</b> .....	<b>5</b>
<b>Solution Description</b> .....	<b>6</b>
<b>Solution</b> .....	<b>7</b>
<b>Conclusions</b> .....	<b>19</b>
<b>About the author</b> .....	<b>20</b>
<b>Document Information</b> .....	<b>21</b>
<b>Appendix A. Additional Tips</b> .....	<b>22</b>

## Table of Figures

Figure 1 Generating script to create F00950_History.....	7
Figure 2 Scripting formatting.....	8
Figure 3 Scripting options .....	8
Figure 4 Previewing script .....	9
Figure 5 Modified table creation script.....	10
Figure 6 Testing new table F00950_History .....	11
Figure 7 Creating trigger .....	12
Figure 8 Trigger creation window.....	13
Figure 9 Insert trigger creation syntax check and trigger creation .....	14
Figure 10 Testing insert trigger.....	15
Figure 11 Testing insert trigger.....	15
Figure 12 Testing update trigger .....	17
Figure 13 Testing insert trigger.....	17
Figure 14 Testing delete trigger.....	18
Figure 15 Testing delete trigger.....	18

## Executive Summary

Sarbanes-Oxley imposes new requirements on JD Edwards EnterpriseOne customers, particularly in the areas of tracking changes to the system. SOX's focus on the area of division of responsibilities makes being able to produce documentation or an audit trail of changes to security important to passing your Sarbanes review.

Having a full and complete history of application security changes is a valuable troubleshooting tool for administrators. Knowing what security was in place at a certain time can aid in determining the root cause of system issues and the information contained in the security history table can be used to roll security changes back to a previous configuration if necessary.

Whatever use you have for keeping a record of F00950 security changes you can agree that maintaining the information is good practice, takes little space and has an almost negligible impact on system performance. Spending thirty minutes to set up this method of EnterpriseOne security history tracking could well save you untold hours in the future.

## **Abstract**

The purpose of this paper is to describe the methods used to create an EnterpriseOne security history table that maintains a list of all application security inserts, updates, and deletes.

Step-by-step instructions, including screen captures detailing the setup are included to enable the EnterpriseOne system administrator to easily configure their system to track security changes for the following security types that are maintained in F00950: Application Security, Action Security, Row Security, Column Security, Processing Option Security, Tab Security, Exit Security, Exclusive Application Security, External Calls Security, Solution Explorer Security, Portal Security. All security changes including new records (inserts), updates to existing records (updates), and importantly, deletion of records (deletes) are tracked.

A bonus section is included with additional tips.

Total Pages - 22

## **Problem**

In EnterpriseOne, by default a method of tracking complete history of application security changes does not exist. Only the last modified record is kept in the F00950 table. No true audit trail exists for inserts, updates or deletions. While the F9312 table can be used to track the record of a security change, it does not record the actual change, just the fact that the change took place.

The EnterpriseOne administrator is faced with the prospect of not knowing what security changes have taken place in the past and only knowing what the last security change was for a certain object.

## **Solution Description**

Creating a method of keeping track of all application security changes to the EnterpriseOne system involves the creation of a "history" table to which records are written when actions are taken in the P00950 Application Security program. The records include the same information as is in the original F00950 plus columns to keep track of the date the transaction occurred and to notate the type of action (insert, update, delete) that was taken.

The method of creating these records is SQL Server triggers, which fire when certain events occur in the table the trigger is created for. When these events occur, the trigger is designed to take certain actions, in our case write a record. These records get written to the F00950\_History table and, as stated earlier, include all information that may be needed in the future.

## Solution

### Setup – Table Creation

To begin, we must create the table that the history records will be stored in. Obviously, since we wish to keep the same information that is currently in the F00950, that table is a prime candidate on which to base our history table.

The easiest way to create the history table is to use a SQL Serve method called "scripting an object" where one generates sql commands that can be used to re-create the selected object, in this case F00950. We are going to generate the F00950 script, modify it, and use it to create our F00950\_History table.

Below we begin the process by opening SQL Server Enterprise Manager, selecting SYS7334.F00950 in the JDE7334 database , right-click and choose "All Tasks/Generate SQL Script".

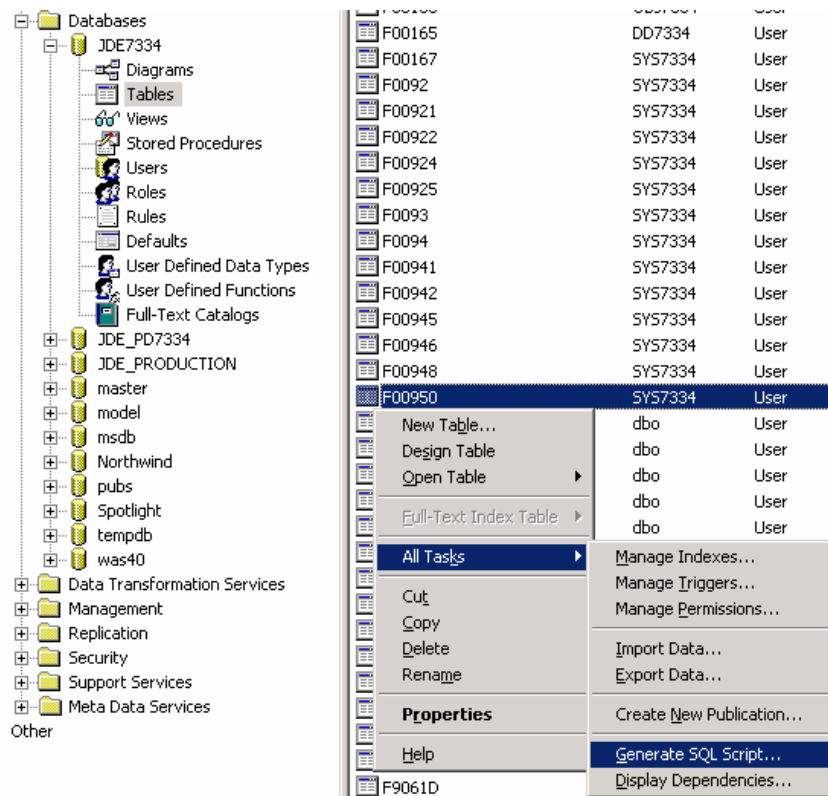


Figure 1 Generating script to create F00950\_History

This will open the Generate SQL Scripts window where we will change a few options. Begin by unchecking the "Generate the DROP <object> command for each object" option.

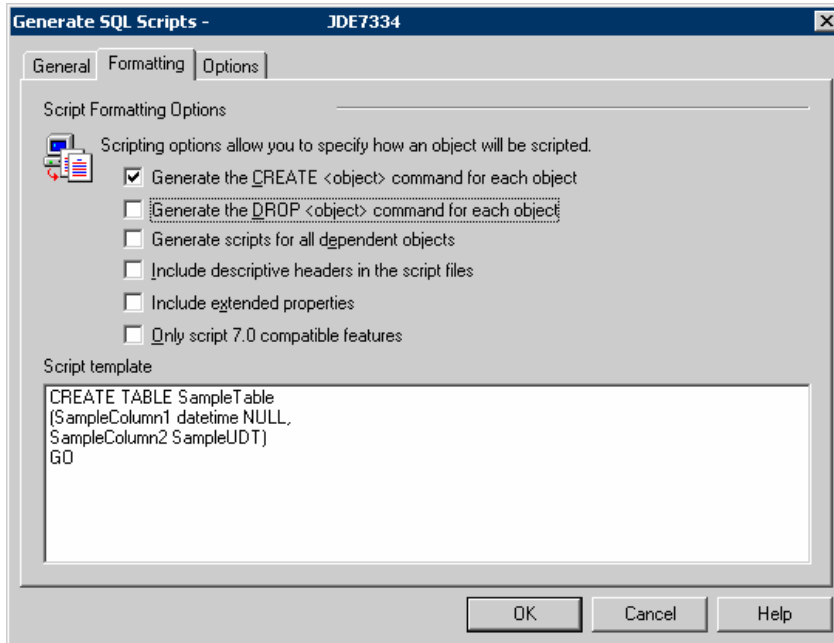


Figure 2 Scripting formatting

Select the "Options" tab and check the "Script object-level permissions" box.

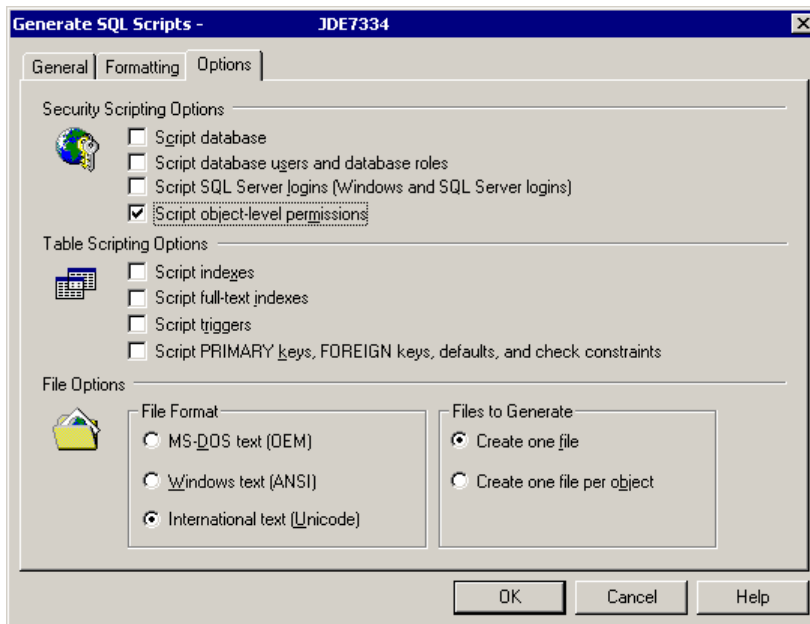


Figure 3 Scripting options

Select the "General" tab and click the "Preview" button to produce the script.

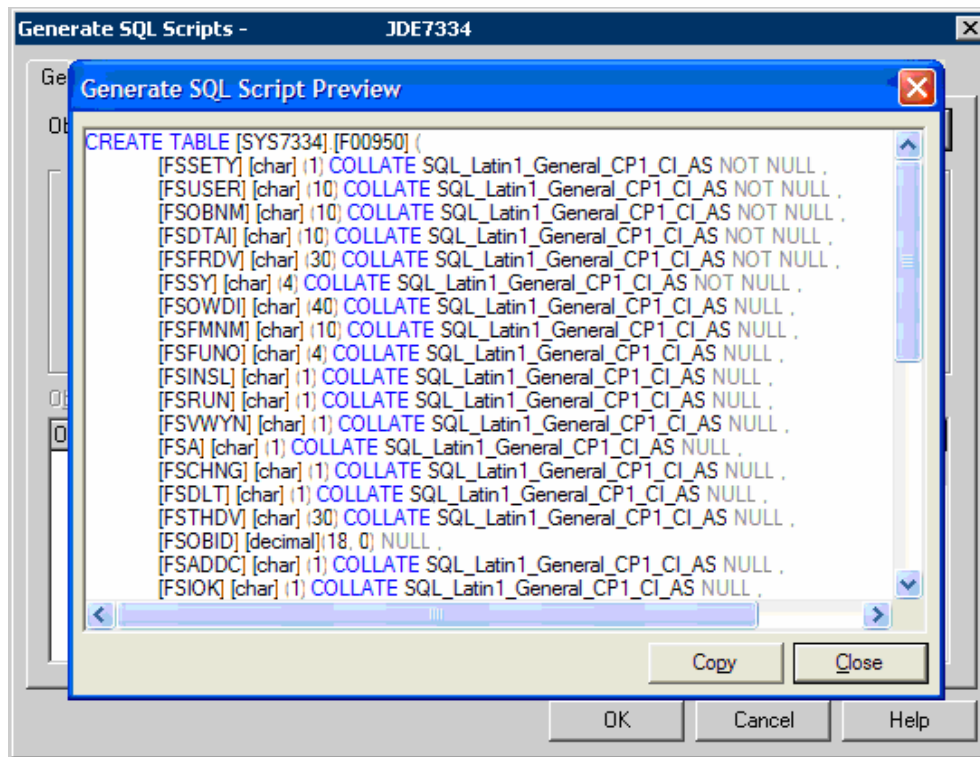


Figure 4 Previewing script

This script is going to become the basis for the creation of our F00950\_History table. After a few modifications we will run this script in SQL Query Analyzer. Copy the script text from the "Generate SQL Script Preview" window.

Open SQL Query Analyzer and paste the script text in. Modify the sections highlighted in red below, changing the table name from **F00950** to **F00950\_History** and adding a comma after the last statement in the original table creation script. Then add **[modified\_date] [datetime]** and **action [varchar] (15)**. These are the additional columns that are added to the history table. Lastly, change the permissions statement to **F00950\_History** instead of **F00950**.

Execute the script.

```

CREATE TABLE [SYS7334].[F00950_History] (
  [FSSSETY] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [FSUSER] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [FSOBMM] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [FSDTAI] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [FSFRDV] [char] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [FSSY] [char] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
  [FSOWDI] [char] (40) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSFMNM] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSFUNO] [char] (4) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSINSL] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSRUN] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSVWYN] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSA] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSCHNG] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSDLT] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSTHDV] [char] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSOBID] [decimal](18, 0) NULL ,
  [FSADDC] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSIOK] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSICPY] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSIUPT] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSATN1] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSATN2] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSATN3] [char] (1) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSMUSE] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSPID] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSJOBN] [char] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
  [FSUPMJ] [decimal](18, 0) NULL ,
  [FSUPHT] [float] NULL ,
  [FSEXITID] [decimal](18, 0) NULL ,
  [FSTABID] [decimal](18, 0) NULL ,
  [FSTEXTID] [decimal](18, 0) NULL ,
  [FSPTH] [char] (120) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
  ,
  [modified_date] [datetime] ,
  [action] [varchar] (15)
) ON [PRIMARY]
GO

GRANT REFERENCES , SELECT , UPDATE , INSERT , DELETE ON [SYS7334].[F00950_History] TO [public]
GO

```

Figure 5 Modified table creation script

## Setup – Table Creation Testing

The next step tests our newly created table. Open a new Query Analyzer session. Enter the following query and execute: **Select \* from sys7334.f00950\_History**

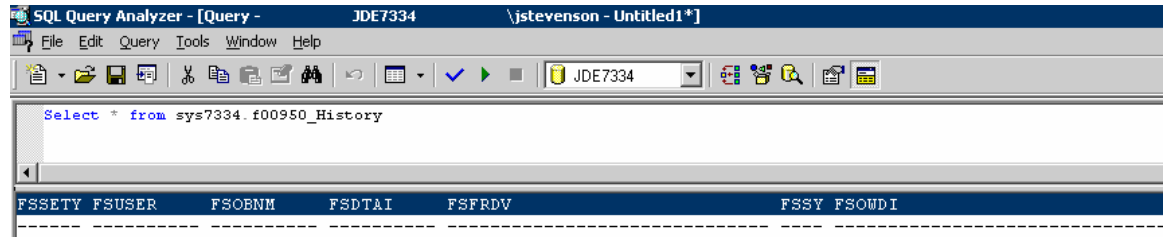


Figure 6 Testing new table F00950\_History

If your table creation script worked you should get no errors and results in the bottom that contain no records but will list the field names

## Setup – Trigger Creation and Testing

Return to Enterprise Manager and again select the SYS7334.F0090 table in JDE7334 database, right-click and select All Tasks/Manage Triggers.

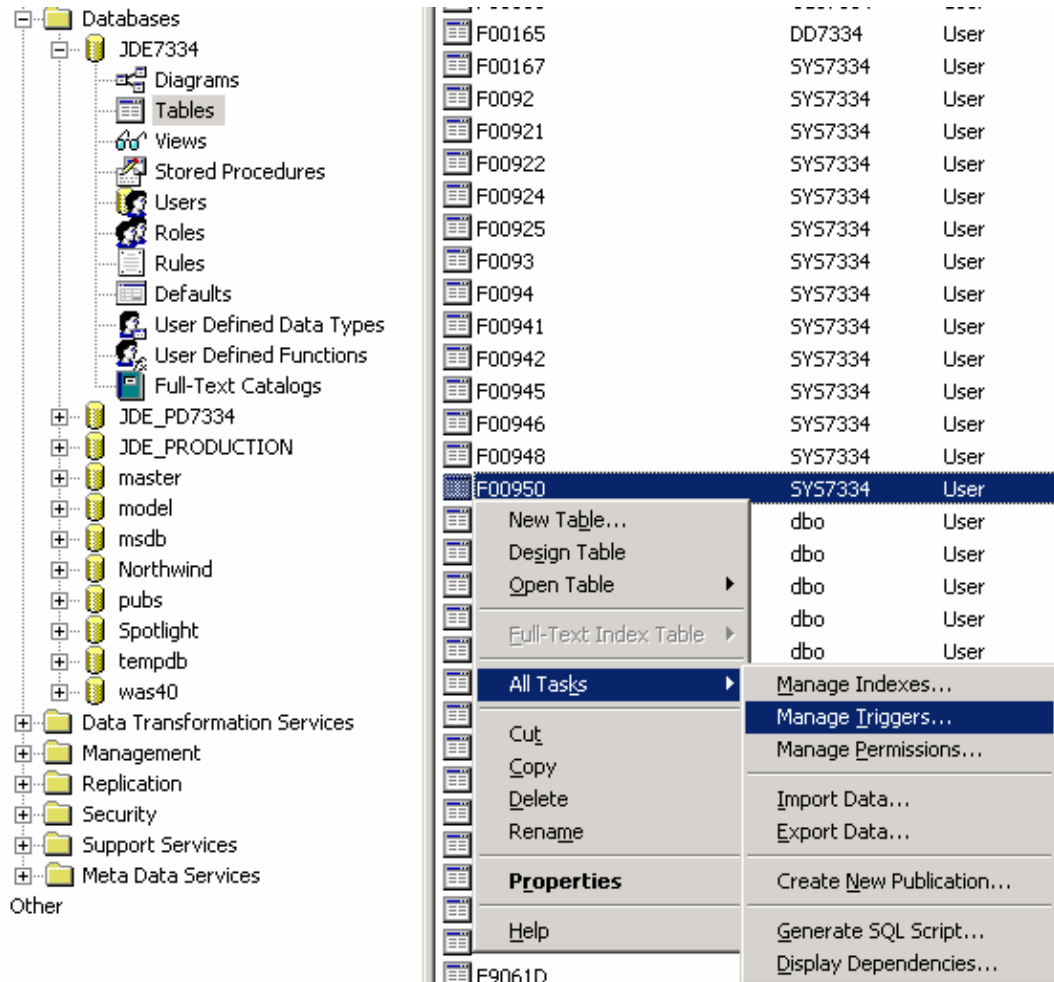


Figure 7 Creating trigger

This will bring up the Trigger Properties window.

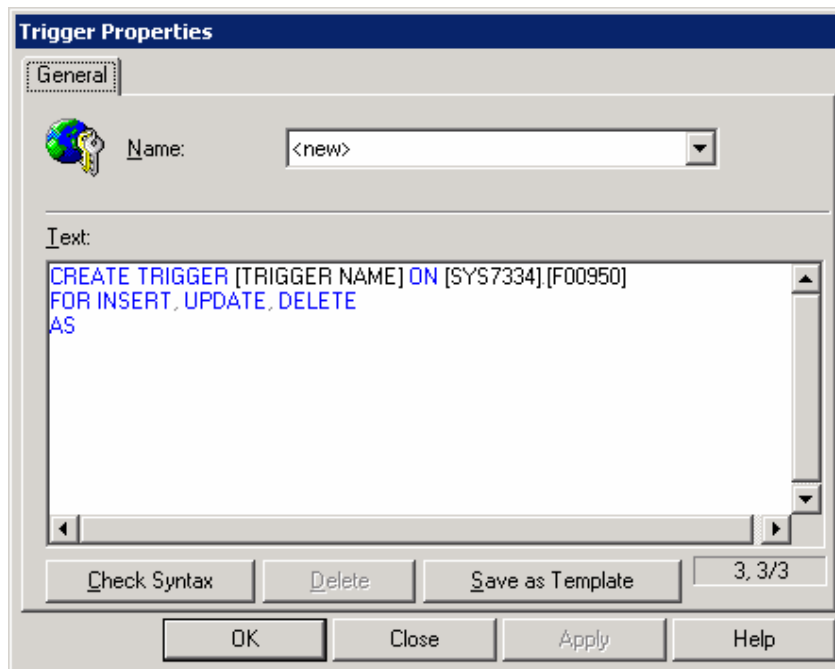


Figure 8 Trigger creation window

Replace the text in the window with the following text to create the insert trigger:

```
CREATE TRIGGER [trg_InsertHistory] ON [SYS7334].[F00950]
```

```
FOR INSERT
```

```
AS
```

```
Insert SYS7334.F00950_History (FSSETY, FSUSER, FSOBNM, FSDTAI, FSFRDV,  
FSSY, FSOWDI, FSFMNM, FSFUNO, FSINSL, FSRUN, FSVWYN, FSA, FSCHNG,  
FSDLT, FSTHDV, FSOBID, FSADDC, FSIOK, FSICPY, FSIUPT, FSATN1,  
FSATN2, FSATN3, FSMUSE, FSPID, FSJOB, FSUPMJ, FSUPMT, FSEXITID,  
FSTABID, FSTEXTID, FSPTH, modified_date, action)
```

```
Select FSSETY, FSUSER, FSOBNM, FSDTAI, FSFRDV, FSSY, FSOWDI,  
FSFMNM, FSFUNO, FSINSL, FSRUN, FSVWYN, FSA, FSCHNG, FSDLT,  
FSTHDV, FSOBID, FSADDC, FSIOK, FSICPY, FSIUPT, FSATN1, FSATN2,  
FSATN3, FSMUSE, FSPID, FSJOB, FSUPMJ, FSUPMT, FSEXITID, FSTABID,  
FSTEXTID, FSPTH, GETDATE(), 'INSERTED'
```

```
From inserted
```

Click the "Check Syntax" button to ensure the code is correct, then click the "Ok" button

to create the trigger.

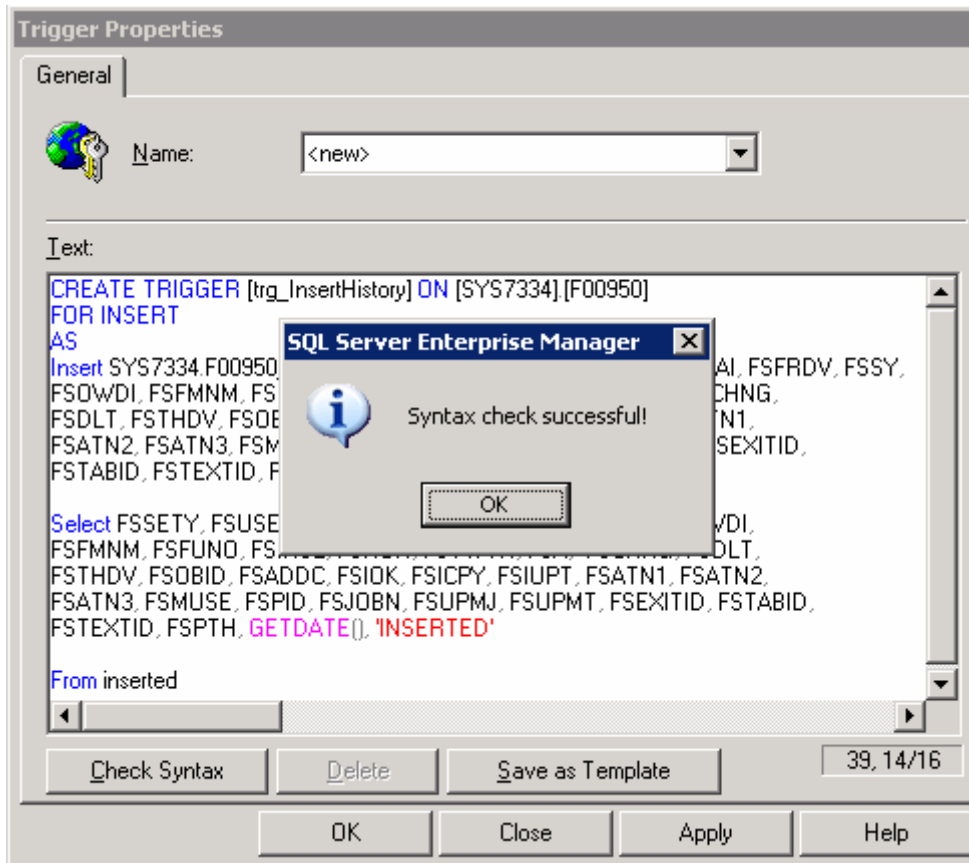


Figure 9 Insert trigger creation syntax check and trigger creation

Open EnterpriseOne , enter the Work With Application Security application (P00950) and create a new security record.

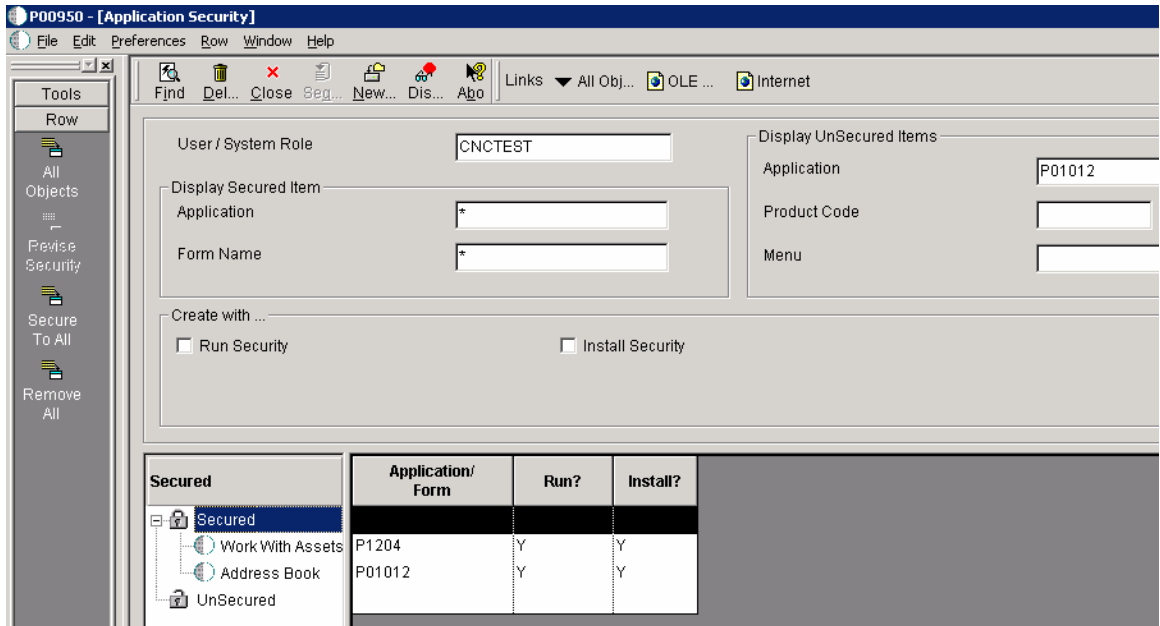


Figure 10 Testing insert trigger

Return to Query Analyzer and again run the query **Select \* from sys7334.f00950\_History**. The results should show a single record, the copy of the security record created in the previous step.

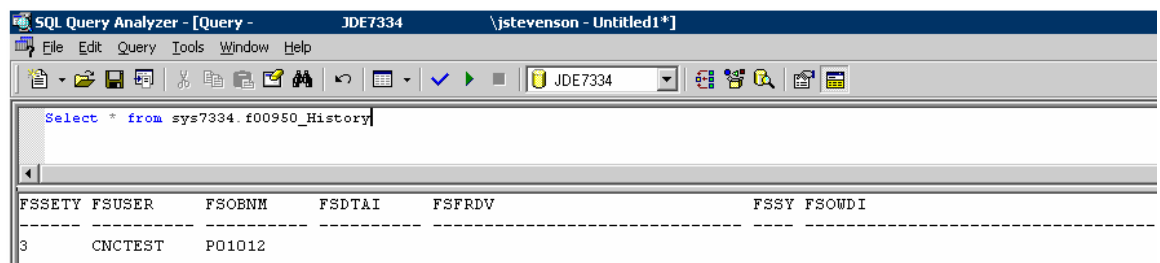


Figure 11 Testing insert trigger

Continue on and create two more triggers, the triggers for updates and for deletes:

```
CREATE TRIGGER [trg_UpdateHistory] ON [SYS7334].[F00950]
FOR UPDATE
AS
Insert SYS7334.F00950_History (FSSETY, FSUSER, FSOBNM, FSDTAI, FSFRDV, FSSY,
FSOWDI, FSFMNM, FSFUNO, FSINSL, FSRUN, FSVWYN, FSA, FSCHNG,
FSDLT, FSTHDV, FSOBID, FSADDC, FSIOK, FSICPY, FSIUPT, FSATN1,
FSATN2, FSATN3, FSMUSE, FSPID, FSJOBN, FSUPMJ, FSUPMT, FSEXITID,
FSTABID, FSTEXTID, FSPTH, modified_date, action)

Select FSSETY, FSUSER, FSOBNM, FSDTAI, FSFRDV, FSSY, FSOWDI,
FSFMNM, FSFUNO, FSINSL, FSRUN, FSVWYN, FSA, FSCHNG, FSDLT,
FSTHDV, FSOBID, FSADDC, FSIOK, FSICPY, FSIUPT, FSATN1, FSATN2,
FSATN3, FSMUSE, FSPID, FSJOBN, FSUPMJ, FSUPMT, FSEXITID, FSTABID,
FSTEXTID, FSPTH, GETDATE(), 'UPDATED'
From inserted
```

```
CREATE TRIGGER [trg_DeleteHistory] ON [SYS7334].[F00950]
FOR DELETE
AS
Insert SYS7334.F00950_History (FSSETY, FSUSER, FSOBNM, FSDTAI, FSFRDV, FSSY,
FSOWDI, FSFMNM, FSFUNO, FSINSL, FSRUN, FSVWYN, FSA, FSCHNG,
FSDLT, FSTHDV, FSOBID, FSADDC, FSIOK, FSICPY, FSIUPT, FSATN1,
FSATN2, FSATN3, FSMUSE, FSPID, FSJOBN, FSUPMJ, FSUPMT, FSEXITID,
FSTABID, FSTEXTID, FSPTH, modified_date, action)

Select FSSETY, FSUSER, FSOBNM, FSDTAI, FSFRDV, FSSY, FSOWDI,
FSFMNM, FSFUNO, FSINSL, FSRUN, FSVWYN, FSA, FSCHNG, FSDLT,
FSTHDV, FSOBID, FSADDC, FSIOK, FSICPY, FSIUPT, FSATN1, FSATN2,
FSATN3, FSMUSE, FSPID, FSJOBN, FSUPMJ, FSUPMT, FSEXITID, FSTABID,
FSTEXTID, FSPTH, GETDATE(), 'DELETED'
From deleted
```

Return to EnterpriseOne and modify the security record you created earlier. This will test the update trigger.

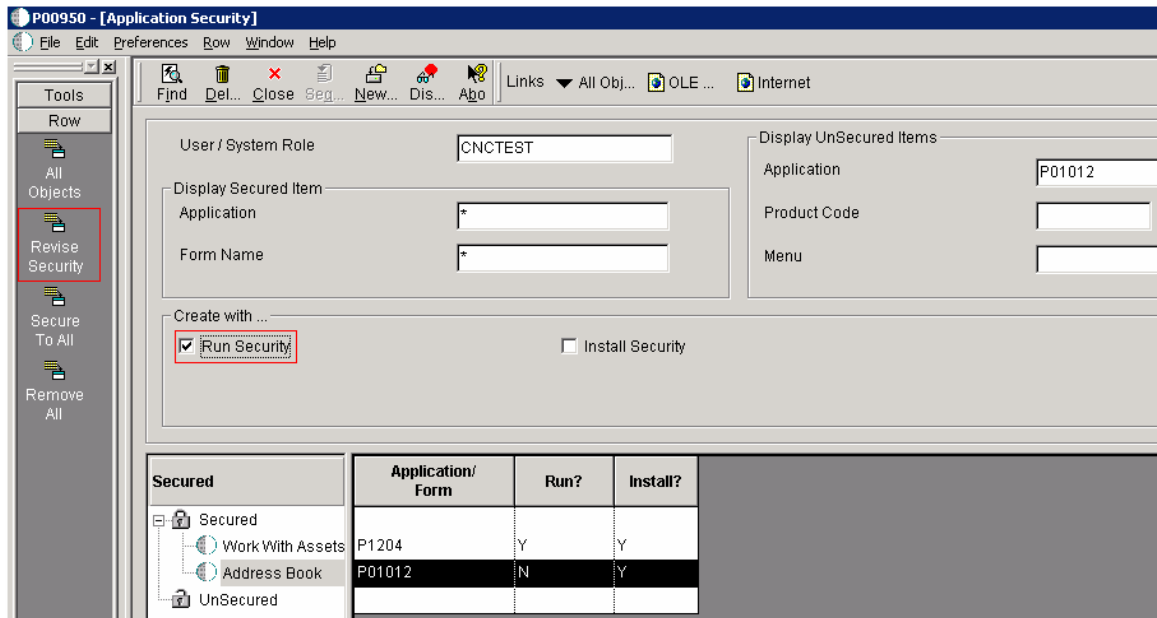


Figure 12 Testing update trigger

Return to Query Analyzer and again run the query **Select \* from sys7334.f00950\_History**. The results should now show two records, the copy of the insert record and now a copy of the update record. Note that in the column "Action" we see the two actions taken thus far.

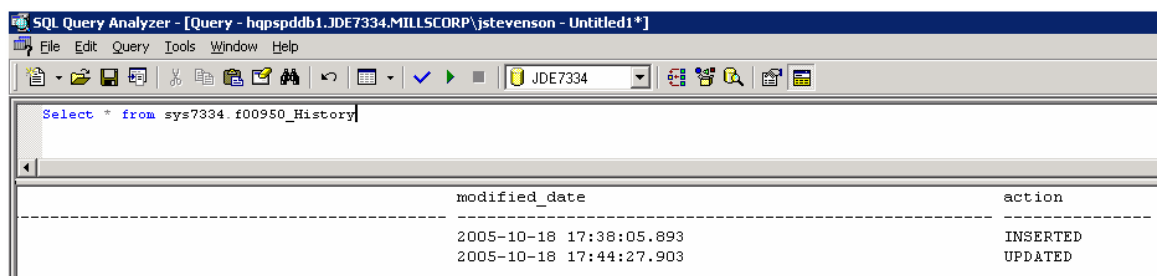


Figure 13 Testing insert trigger

We are now going to test the last function by deleting the created and modified security record. Select the security record and delete it from P00950.

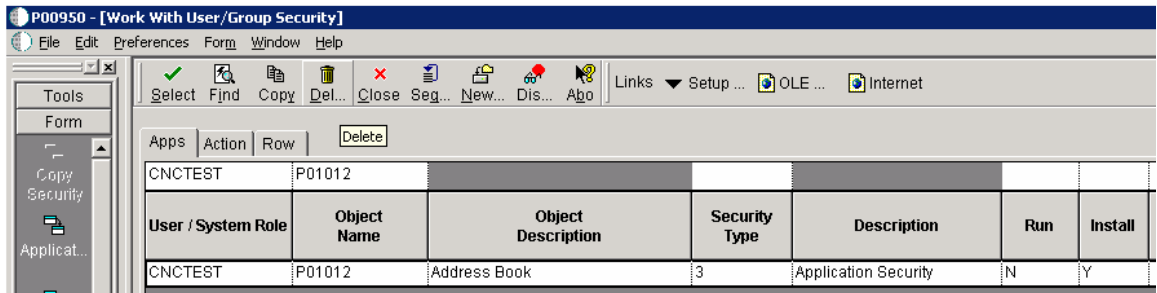


Figure 14 Testing delete trigger

Return to Query Analyzer and run the query **Select \* from sys7334.f00950\_History**. The results should now show three records, the two previous records plus the record that shows a delete took place in the F00950 table.

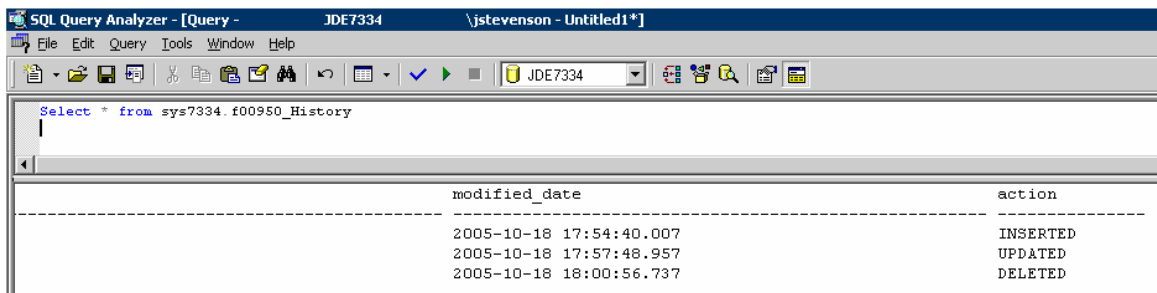


Figure 15 Testing delete trigger

From now on your system will record all actions taken in the P00950 Application Security app in the F00950\_History table. The records in the F00950\_History table will be exact copies of the added, changed, or deleted records in the F00950 table. The records in the F00950\_History table will also contain information on what action was taken on the record and when it took place, giving you a thorough trail of Application Security activity in your EnterpriseOne system.

## Conclusions

Using the methods provided in the solution section, an administrator will be able to create the structures necessary to track EnterpriseOne application security changes.

Doing so will give the system administrator a database table tracking all application security changes to their EnterpriseOne system, useful for meeting Sarbanes-Oxley audit requirements and for troubleshooting system issues related to security changes.

## About the author

Jeff Stevenson is a senior technologist with broad experience on SQL Server, WebSphere, Windows, Linux, Apache, Drupal and other technologies but primarily focusing on Oracle's JD Edwards EnterpriseOne product line.

Jeff has over six years of award-winning EnterpriseOne experience and has spent fifteen years in the high-tech world, including a stint in the US Marine Corps working on computer and electronics systems of advanced fighter jets.

A leading member of the EnterpriseOne community, he has given numerous presentations, conducted workshops and technology survival camps both as an employee of JD Edwards/PeopleSoft and independently. He is an active participant on the [jdelist.com](http://jdelist.com) website, dispensing advice under the pseudonym Brother of Karamazov.

Jeff lives near Washington, DC in Gaithersburg, Maryland with his wife and two girls and likes to participate in short triathlons when not hovering over the keyboard.

---

## Document Information

<b>Title</b>	E1, Tracking Application Security Changes in EnterpriseOne
<b>Author</b>	Jeff Stevenson
<b>File Name</b>	E1, Tracking Application Security Changes in EnterpriseOne.doc
<b>Revision</b>	1.10
<b>Technology 1</b>	SQL
<b>Technology 1 Version</b>	SQL 2000/7.0
<b>Technology 2</b>	EnterpriseOne
<b>Technology 2 Version</b>	All
<b>Technology 3</b>	
<b>Technology 3 Version</b>	
<b>Price</b>	

## Appendix A. Additional Tips

---

- To ensure that your new security history table does not grow wildly be sure to place a reference to it in some kind of periodic maintenance chart and archive.
- You can use the same trigger method to keep track of the submitted jobs records. If you clear your submitted jobs list periodically you lose the history. By maintaining a F986110\_History table you can delete the WSJ records (which delete the PDF's) and still keep a record of what jobs were run and the accompanying job information.
- You can also use this method to keep track of OCM changes.