Building an Audit Trail in an Oracle Applications Environment

Abstract: (300)
Sarbanes-Oxley's section 404 requires a company's key systems be audited. However, many companies have 'unauditable' systems and don't even know it. This paper explores methods by which companies can create an auditable system by implementing various levels of audit trails in Oracle Applications.

Executive Summary: (4000)
The technology revolution of the past ten years has been staggering. We have witnessed monumental and historic events such as Y2K and the dot-com boom/bust. As part of this revolution, we have seen many companies migrate from using mainframe systems to client-server and, most recently, to web-enabled applications. We have also observed the explosion of ERP applications that run on multi-tier architectures. With new technology brings new challenges.

Alongside the revolution of technology, there has been a revolution of corporate accountability. Legislation such as Sarbanes-Oxley (SOX), California Senate Bill 1386, HIPAA, Basel II, and the Gramm-Leach-Bliley Act (among others) has resulted in greater scrutiny and larger penalties for poor corporate governance and security. Much of the risk from this legislation rests on IT systems.

Sarbanes-Oxley's section 404 requires a company's key systems to be audited. However, many companies have 'unauditable' systems and don't even know it. There is no magic solution or silver bullet in creating an auditable system that satisfies all the requirements for proper corporate governance; rather a layered approach of auditing and logging is required to provide a comprehensive and thorough audit trail. This paper explores methods and degrees by which companies can create an auditable system by implementing various levels of audit trails in Oracle Applications.

There are five primary ways to develop an audit trail:

1. Standard Application Auditing
2. Application Level Audit Trail
3. Database Event Auditing
4. Database Trigger Auditing
5. External Auditing

Standard Application Auditing comes with the 'vanilla' install and provides the last update who and last update when information. However, throughout most of the application, detailed tracking of the changes made between the initial creation of a record and the last change made doesn't exist, but is necessary to show to an auditor. Therefore, this method, alone, is not a sufficient audit trail. Unfortunately, for most companies that run Oracle Applications, this is the only audit trail they have (although some applications and processes do have an audit trail built into their development).

Application level audit trail allows a company to track changes at the detailed level. When enabled at the table level through the Sys Admin function, the application works with the database to build detailed records on changes, additions, and deletions at the database level. This audit trail is stored in a separate table from the production data, which still maintains the Standard Application Auditing discussed above. This is part of the solution for companies, but is rarely enabled by companies due to the concern about system performance.

Database event auditing tracks activity at the event level. An example of an event is the user logging into the applications. Database event auditing is necessary to track certain events, but is rarely enabled by companies.

Database trigger auditing is the alter ego of the Application Level Audit Trail as that feature relies on database triggers to form the audit trail. Database trigger auditing may be necessary to record changes made at the database level, not made through the application login. Some form of database trigger auditing is necessary, but is rarely enabled by companies.

External auditing allows an audit trail to build through the 'mining' of certain information contained in the redo logs. This approach gives access to the greatest amount of data in that it allows for the mining of any change or access of the data. However, it is next to impossible to build internally and very expensive to purchase the technology from a third party vendor.

The net result is that a company's need to develop a comprehensive strategy using various technologies crafted to their circumstances. Absent an audit trail, a company's applications may not be 'auditable' and an auditor could assess a material weakness or significant deficiency in the context of a company's section 404 internal controls report.

White Paper

The technology revolution of the past ten years has been staggering. We have witnessed monumental and historic events such as Y2K and the dot-com boom/bust. As part of this revolution, we have seen many companies migrate from using mainframe systems to client-server and, most recently, to Web-enabled applications. We have also observed the explosion of (Enterprise Resource Planning) ERP applications that run on multi-tier architectures. With new technology brings new challenges.

Alongside the revolution of technology, there has been a revolution of corporate accountability. Legislation such as Sarbanes-Oxley (SOX), California Senate Bill 1386, HIPAA, Basel II, and the Gramm-Leach-Bliley Act (among others) has resulted in greater scrutiny and larger penalties for poor corporate governance and security. Much of the risk from this legislation rests on IT systems.

In this white paper, we will explore the challenges companies face in developing an auditable system given these new technologies and the increased emphasis on corporate governance and security. We will also explore methods and degrees by which companies can address these challenges by implementing various levels of audit trails in their ERP applications.

Why is an audit trail so important – developing context

Sarbanes-Oxley (SOX) legislation passed in 2002 has had the most far-reaching impact of all the corporate governance initiatives. More recently auditors who are auditing under SOX's 404 provisions are placing greater emphasis on general IT controls surrounding the 'key systems' identified as part of a company's controls assessment. The audit of a company's controls has turned into a "de facto audit of IT" (Source: CFO IT, Summer 2005). Auditors are changing their approach to the audit of a company's change management process from only testing a few change requests to testing for unauthorized changes. To do so auditors are requesting (and sometimes requiring) companies to prove that all system changes have been properly authorized and processed according to their change management plan. To facilitate this depth of a review, companies need to be able to identify all 'system' changes to their applications, at least for the systems being audited under SOX's 404 provisions. Unfortunately, implementers and IT management have had little focus on developing strong general IT controls in conjunction with the implementation and development of these new technologies. This leaves many companies exposed to the *possibility* of having an 'unauditable' system because many companies cannot identify all changes made to a system. Some applications and processes do have a standard audit trail. An example of an application is HRMS that allows for date tracking of changes. An example of a process that has a standard audit trail is purchasing which can be implemented to record the approval hierarchy for a given purchase order.

Having an unauditable system will lead to various challenges. One example is the automation of controls. Many companies have put a lot of effort into the automation of controls. If the general IT controls cannot be relied upon (i.e. change management

process), the various application controls may not be able to be relied upon or may require significant testing to validate them. If a proper audit trail for testing of a company's change management process cannot be presented to your auditor, they will doubt, at least to some extent, the effectiveness of your change management process. At the extreme, they may disallow the use of your company's application controls.

What are 'system' changes

In an Oracle Applications environment there are several layers to the system. There is 'code' stored at the OS level, database level, and at the middle tier. A change to any of the code (also referred to as objects) should be subject to a company's change management plan, regardless how small or large the change. Code changes also include the change of some of the setups within the application that can be done through the application (user interface/forms). Many setups have the effect of changing code because they cause the application to process/react differently based on the different configurations. A sample list of 'system' changes that would need to have an audit trail would include:

- Changes to the database structure
- Addition, deletion, or change to database triggers
- Changes to programs, libraries, or scripts at the OS level
- Changes to objects or packages at the database level
- Changes to the setups or profile options at the application level

How then is a company to develop an audit trail of all changes made at each of these three levels? This paper will focus primarily on the changes made at the application and database levels.

As we will see as we explore this topic further, a strategy to develop an audit trial will require a company to thoughtfully put together a strategy with people from various skill sets throughout its' organization. Development of the strategy often requires input from the internal audit department, your external auditors or Sarbanes-Oxley advisors, the IT security team, functional users in Finance and other departments, the application DBAs, and application system administrators.

Unfortunately, there is no magic solution or silver bullet in creating an auditable system that satisfies all the requirements for proper corporate governance. Rather a layered approach of auditing and logging is required to provide a comprehensive and thorough audit trail. There are five primary ways to develop an audit trail:

1. Standard Application Auditing
2. Application Level Audit Trail
3. Database Event Auditing
4. Database Trigger Auditing
5. External Auditing

Standard Application Auditing

Oracle Applications by default provides only basic auditing functions. The application automatically tracks standard Who/When for most records as it relates to creating

(CREATED_BY and CREATION_DATE) and updating (LAST_UPDATED_BY, LAST_UPDATED (date function)). The underlying tables supporting the application store who/when the record was created and who/when the record was last updated. However, in most cases, the database doesn't store WHAT was changed on the record. There are some exceptions to this in certain applications like HR/Payroll where 'date-tracking' is critical, but this is predominantly the case throughout most of the applications. ***Without detailed tracking of the changes made to a record, there is not a sufficient audit-trail for an auditor to review.***

Having an audit trail only at the application level does not provide the critical 'what' component required by a proper audit trail. If an auditor wants to review a change to setups, for example and all that is stored in the database is the when/who of the record creation and the when/who of the record last update, it doesn't provide any visibility to the changes that could have been made been the first time it was created and the last time it was updated. Furthermore, it doesn't provide the visibility to what column, in particular, was changed. 'What' was changed has a significant impact on the risk of the change.

Also, the standard application audit trail allows you to audit when a user logs in, what responsibilities or roles they use and what forms they access; however, this auditing is not enabled by default. To enable logging of all user, responsibility, and forms accesses, set the system profile option 'Sign-On: Audit Level' to FORMS (default is NONE).

Most commonly in the implementation of Oracle Applications, only the default application level of audit trail is the extent of most companies' audit trail. To the degree that this is still the case for most companies, ***significant audit risk still exists***. Companies need to remedy this as soon as possible.

Pros of Application Level auditing:

1. Standard part of the application configuration
2. No performance impact

Cons of Application Level auditing

1. Does not provide detail on what gets changed in most cases – just who and when a record is updated
2. Does not provide the level of detail required by an auditor
3. Does not track enough detail to be able to reconstruct activity in the case of fraud

Application Level Auditing

Oracle Applications has the ability to build a complete audit trail of changes made through the application. When a user logs into the application, certain information is maintained by the application about that session, including the application login used. Application level auditing works with the database to record information about a transaction as the application interacts with the database. For those tables/columns that are configured to do so, as records are changed (added, deleted, or updated) in these

tables, a database trigger replicates the change to a mirror table, thereby providing a complete, detailed audit trail of all changes including who, when, and what.

This function is enabled through the System Administrator seeded responsibility. Development of an audit trail through the application level begins with the identification of specific tables and columns that a company wishes to audit. However, the downside to this method of developing an audit trail is the impact that triggers have on the performance of the database. The additional overhead required by the trigger makes auditing tables with high transactional volume impractical, if not impossible.

Let's look at an example. The ABC Company has determined it wants to audit the update of the value in the chart of accounts so it has an audit trail of every change made. It identifies that the values related to the chart of accounts are stored in the FND_VALUES table. Because the volume of transactions hitting the FND_VALUES table isn't significant, it decides to audit all activity in that table and does so by setting it up via the System Administrator's Audit Trail menu. The next time that a user inserts a value to the ABC_ACCOUNT value set, when the record is saved, the application passes certain session information (i.e. the user login) to the database with the change in the data. The database trigger, having been enabled to develop an audit trail, fires and records this data, along with the user id, in a mirror table. The core application table FND_VALUES only stores the last updated who, last updated (when) values. It doesn't store what was updated and, therefore, this special "Application Level Auditing" is necessary to develop the full audit trail.

ERP systems, like Oracle Applications, have caused a dramatic rise in the role of what most companies refer to as a Business Analyst or Super User. These employees are typically 'power users' that become the experts in the configuration, development, maintenance, and support of the ERP system. Because of the extremely flexible configurability of ERP systems, users who specialize in this area are sometimes given liberal access to the production environment (although this has changed at many companies because of controls put in place in light of Sarbanes-Oxley).

However, given that these types of users would normally be given access to high volume forms and functions such as PO and AP entry, it would be necessary to audit the tables that support such processes in order to track what these users are doing while having access to these forms. Practically speaking, auditing such tables would cause a significant or perhaps debilitating impact on the performance of the application. (See the white paper written on Super User Best Practices available at www.oubpb.com.)

As a solution to this, many companies have revoked access to such forms for these Super Users or have given the inquiry only access (which may require a significant amount of development effort to accomplish). In smaller companies, where the Super User may also act as the System Administrator (granting access to the applications), significant additional risk accrues (see white paper regarding Sys Admin Challenges in Small Companies available at www.oubpb.com).

Pros of Application Level Auditing:

1. Can provide meaningful auditing of low-volume, high-risk tables such as core application set ups and application security tables (like those contained in the Application Object Library)
2. Part of standard application/database technology
3. Can provide much more detailed audit trail than Standard Application Auditing because all updates, inserts, and deletions can be recorded

Cons of Application Level Auditing:

1. Many companies and DBAs have not enabled Application Level Auditing and are unfamiliar with its' deployment
2. Auditing high volume transactional tables is impractical, if not impossible, due to the impact on performance caused by the database triggers
3. Those with sufficient understanding of, and access to, the audit tables can alter the audit trail

Note: There are also methods of developing an audit trail using workflow, alerts, and the custom library, however, these methods require significant customization to the application. These methods primarily have the same issues as the use of Oracle's standard application level audit.

Database Event Auditing

The Oracle Database has sophisticated built-in auditing capabilities. Many database operations can be audited including database logins, key SQL statements, and data access. The audit trail information can be stored either in the database or to a file.

Since the auditing is enabled at the database level, there is no application user session information captured. All Oracle Applications users' sessions and most maintenance programs connect to the database using a single database account (APPS), therefore, the value of the auditing information is diminished.

Database Event Auditing only captures WHO and WHEN information, not WHAT was changed for many of the audited database operations. Auditing SELECT, INSERT, UPDATE, and DELETE SQL statements provide only information on the database object (table or view) that was accessed, not what data was manipulated.

Database Event Auditing is most useful in capturing accesses to the database and changes in the structure and privileges, rather than auditing changes in individual tables. Since this type of auditing is built-in functionality, it is easy to configure and there is minimal performance impact. The most significant downside is that the audit trail can be easily manipulated by the DBA.

Pros of Database Event Auditing:

1. Part of standard database technology
2. Easy to configure and implement

Cons of Database Event Auditing:

1. Those with sufficient understanding of, and access to, the audit tables can alter the audit trail
2. Application user session information is not captured

Database Trigger Auditing

Database Trigger Auditing is similar to Application Level Auditing and uses the same core technology – database triggers to develop the audit trail. However, database level auditing can audit some transactions that the application cannot. Employees (and unwelcome non-employees) can log directly into the database and use SQL tools to query the data directly instead of accessing the data through the applications. ODBC applications also access the database directly to interact with the database. All interaction with the database requires a login and password just like an application does. So, database level auditing captures changes to the database through logins other than from the application.

There is one other note to mention on Database Trigger Auditing and the difference from Application Level Auditing regarding the interaction between the database and the applications – if auditing is enabled only at the database level, the application session information (e.g., application user name) is not available. All Oracle Applications users' sessions connect to the database using a single database account (APPS). When the database trigger auditing is enabled through the application, the application specific database trigger stores the application user session information in the shadow table. Since the applications connect to the database through the APPS login, if database level auditing alone was enabled the audit trail would show that the APPS user made all the changes in the application. Therefore, as it relates to changes made at the application level, the Application Level Auditing is superior to just using the Database Trigger Auditing.

However, since users and other applications can also directly access the database without logging into the application, it may also necessary to audit certain information at the database level.

Pros of Database Trigger Auditing:

1. Can provide meaningful auditing of low-volume high-risk for database sessions activity on tables such as core application set ups and application security tables (like those contained in FND and SYS schemes, etc.)
2. Part of standard application/database technology

Cons of Database Trigger Auditing:

1. Many companies and DBAs have not enabled application auditing and are unfamiliar with its' deployment. As such, strict best practice project methodology needs to be followed.
2. Auditing high volume transactional tables is impractical, if not impossible, due to the impact on performance caused by the database triggers.

3. Those with sufficient understanding of, and access to, the audit tables can alter the audit trail

External Auditing

The final method we will review is the ability to develop an audit trail external to the database. There are primarily two methods used to capture the audit trail information externally: database redo logs and network traffic. We will focus solely on database redo logs as the redo logs provide a much more complete auditing solution and this capability is delivered with the Oracle database, nevertheless commercial solutions exist that will audit an Oracle database by analyzing network traffic.

Redo logs are used by the database to allow for rolling back of transactions and to allow for recovery of the database in event of a hardware or software failure. The redo logs are a record of all changes in the database, therefore, are the ideal audit trail as every insert, update and delete to the database are captured. Since the redo logs are a core function of the database and are always enabled, there is not the overhead cost of having to create triggers on the tables as with Application Level Auditing and Database Trigger Auditing.

The downside of the redo logs is that the format is cryptic and difficult to understand even for the most expert DBA. There are several solutions to access and query the redo logs to allow for the building of a secure audit trail – Oracle provides the LogMiner tool with the database. LogMiner has limited abilities in version 8i and more functionality in newer database versions such as 9i and 10g (see Metalink note 291686.1). LogMiner is not an auditing solution; therefore, a custom auditing solution must be built if the LogMiner tool is used. Otherwise, several commercial database auditing tools that use the redo logs are available.

Pros of External Auditing:

1. Provides greatest ability to track activity for those that have database access
2. Depending on the architecture of the storage of the data, this audit trail typically cannot be altered by high-risk employees
3. Use of third party tools to extract such data can virtually eliminate risk of corruption of audit trail.

Cons of External Auditing:

1. Depending on the database version and abilities of staff, extracting the correct data via LogMiner can be challenging
2. Purchase of third party tool or use of consultants to develop may be required
3. Additional hardware may be required to support the auditing

Conclusion

Most companies running Oracle Applications have significant audit risk because of the inability to produce the detailed level of audit-trail auditors are coming to expect. With greater scrutiny on IT controls relating to financial systems, companies must take action now by putting in place the technology necessary to build such audit trails.
Unfortunately, there is no magic solution or silver bullet in creating an auditable system

that satisfies all the requirements for proper corporate governance. Rather a layered approach of auditing and logging is required to provide a comprehensive and thorough audit trail.

About the Authors

Jeffrey T. Hare, CPA
Jeffrey T. Hare, CPA is the founder of ERP Seminars (www.erpseminars.com) and the Oracle User Best Practices Board (www.oubpb.com) and has written various white papers on SOX Best Practices in an Oracle Applications environment. He has presented white papers to various users groups throughout the country as well as at OAUG and Appsworld conferences. He is the author and presenter of the seminar "Internal Controls and Security Best Practices in an Oracle Applications Environment." His background includes Big 4 experience, over six years experience in CFO/Controller roles, and working in the Oracle Financials space since 1998. Jeff can be reached at jhare@erpseminars.com.

About ERP Seminars:
We recognize the need for companies to have continuing knowledge of industry Best Practices. We team with respected independent consultants and firms to provide quality, relevant seminars based on these Best Practices prepared and presented by well-rounded professionals with ERP expertise.

About Oracle Users Best Practices Board:
The mission of the OUBPB is the aggregation of willing writers and reviewers who will participate in a process to develop Best Practices for the Oracle community. The end result will be a repository of "best practice" and 'peer-reviewed' white papers and other content for end users and consultants to reference in their projects and ongoing development.

Stephen Kost
Stephen Kost is the Chief Technology Officer for Integrigy Corporation. He has been presenting on Oracle Applications security for the past 5 years and has worked with Oracle Applications since 1992 in many roles including DBA, technical architecture, IT security auditor, and system administrator.

About Integrigy
Integrigy Corporation is a leader in application security for large enterprise, mission critical applications. Our application vulnerability assessment tool, AppSentry, assists companies in securing their largest and most important applications. AppDefend is an intrusion prevention system for Oracle Applications and blocks common types of attacks against application servers. Integrigy Consulting offers security assessment services for leading ERP and CRM applications.