Industry-specific

QAD SOLUTIONS

Manufacturing Applications

Deployment Guide
Audit Trails

QAD

MFG/PRO eB2.1
June 1, 2005

QAD Inc.
6450 Via Real
Carpinteria, California 93013
Phone (805) 684-6614
Fax (805) 684-1890
http://www.qad.com

Contents

Audit Trails Deployment

This document provides an overview of the design of the Audit Trails component of the QAD Enhanced Controls module and describes the impact of auditing on a production MFG/PRO system. It also discusses issues you should consider before implementing auditing and how to apply auditing to customizations.

Introduction

The Enhanced Controls module—specifically the Audit Trails features—for MFG/PRO eB2.1 lets users track changes to all relevant data held in an MFG/PRO database with complete confidence. You can configure the setup of Audit Trails to meet regulatory needs specific to your industry, country, or business.

For MFG/PRO eB2.1, Enhanced Controls is an optional module installed separately from the primary installation. The module includes features that let you manage data capture, reporting, and long-term storage and archiving.

This functionality was developed with extensive involvement of QAD's medical-industry customers, who face changing regulations by the US federal government—in particular due to CFR Part 11, the Food and Drug Administration's regulations for retaining electronic records pertaining to manufacturing of medical devices and drugs.

Significant interest in auditing has also been expressed by customers that are in the process of meeting new regulations imposed by IFRS and Sarbanes-Oxley.

*Installation Guide: Enhanced Control*s describes the steps to take to enable this module in your production database. *User Guide: MFG/PRO eB2.1 New Feature*s contains detailed information on how to set up and use Audit Trails features. This deployment guide is intended to assist in the planning phase, before you begin implementation.

Technology and Design

The Audit Trails functionality has been designed to ensure the integrity of the solution and minimize impact on performance. Fail-safe methods have been created to protect the audit trail from corruption or data loss.

The solution uses the database trigger capability of Progress to ensure that all changes are captured, even when they are not initiated from the MFG/PRO application. Much of the additional audit trail processing is designed to occur in the background, separated from primary user activity. These background processes can have lower priority or be distributed to other CPUs to be scalable and minimize impact on the transaction processing activity of the production system.

When a database table is marked for tracking, any change—create, modify, or delete—to a record in that table invokes a replication trigger that captures the changed data in a staging table. A separate background process—the Audit Trail Creation Process—reads the staging table and stores the changed data permanently in an audit database where it can be accessed for reporting.

The architecture of storing audit trail data in a secondary database was chosen to provide flexibility for managing data that may grow very quickly, depending on the configured use of audit trails. This approach supports multiple historical audit databases that can be online all the time or kept offline and brought online only when necessary.

Capturing Audit Data

Audit trail records are first captured and stored temporarily in a staging table (attmp_mstr) while transactions are occurring in the MFG/PRO production database, QADDB. These records are captured with as little impact as possible to the originating transactions. They remain in the production database until moved to the audit database. This is described in “Storing Audit Trails Permanently” on page 6.

The audit trail records are captured through the use of Progress replication triggers. This design ensures that audit trails are captured whether changes to a table are made from MFG/PRO, some other application, or even from the Progress editor.

Note Audit trails are not captured when changes to the database are made through the Progress SQL-92 engine or through ODBC or JDBC connections, such as connecting to the database from Microsoft Excel. If this is a risk in your environment, you can eliminate it by using Progress database security to revoke update access to tables in the database using these non-4GL tools. See the Progress Database Administration documentation for instructions on using Progress database security. MFG/PRO is delivered without any restrictions to updating the database from the Progress Editor, SQL-92, ODBC, or JDBC.

The module includes administrative functions for specifying database tables to track including an effective date and the time line for the usage of each audit databases. All audit trail setup data resides in the standard QADDB database.

Audit Database

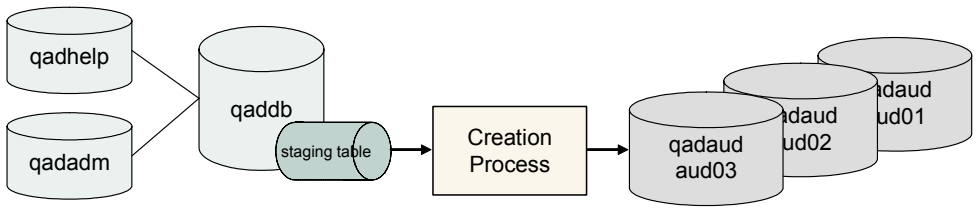
A new audit database—referred to as QADAUD—has been added to the MFG/PRO set of normal production environment databases to store the audit trail data permanently. It has been designed so that several historical audit databases can be available for reporting in addition to the audit database currently used for storing audit events.

The maximum number of records that can be stored in a Progress storage area is two billion, and a table cannot be split between multiple storage areas. For this reason, each of the three tables in the audit database has been allocated its own storage area; each index on each of these tables has been allocated its own storage area as well. This supports storing the maximum number of records in each table.

Because of this finite table capacity and the long-term reporting requirements for the audit trail data, additional audit databases will be needed over time. On any given day, new audit trail data is written to only one audit database. For audit trail reports, multiple audit databases can be accessed based on the date range specified.

Figure 1 shows a typical configuration for a standard MFG/PRO implementation.

Figure 1
Typical Single-Database Audit Implementation

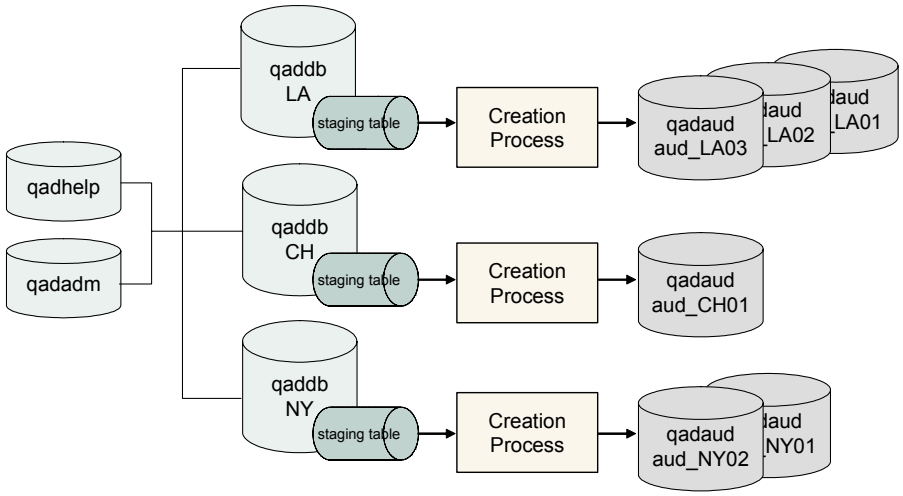


Each MFG/PRO system (production database) has one set of audit databases, regardless of how many domains are in the system. Each audit database stores audit trail data for all domains.

The audit databases can reside on a different machine from the Production database.

Figure 2 illustrates audit databases in an MFG/PRO multi-database implementation.

Figure 2
Typical Multiple-Database Audit Implementation



This figure illustrates how each production database (qaddb NY, qaddb CH, qaddb LA) must have its own corresponding set of audit trail databases. For example, the production database for New York (qaddb NY) has two audit trail databases (aud_NY01 and aud_NY02).

Administering Audit Databases

Audit databases are designed to fill up and eventually be taken offline. You set the date when a new audit database comes online and replaces the old audit database.

Audit information is always written to the current audit database. However, audit reports can be generated against past and current databases. Therefore, the administrator's tasks include maintenance of past, current, and future databases, and the connections to these.

Using the example in Figure 3, assuming that several audit periods have passed, you would need access to all audit databases that may need a client connection. In the example, this would be databases AT02 through AT06. This includes all databases in the reporting horizon as well as the audit database that will take the place of the current one on its effective date.

Database AT06 must exist and have a server running on the day the switchover is assigned in MFG/PRO.

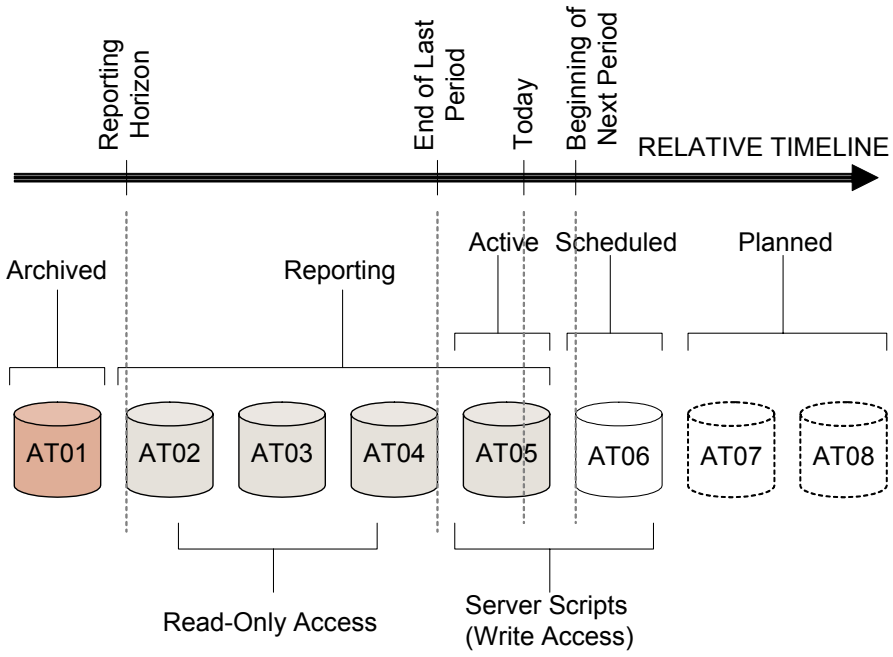


Figure 3
Audit Database Generations

Depending on how often your organization chooses to maintain the server scripts and databases, you may want to create several future audit databases—as well as several server scripts—to address different periods. For example, the next server script might launch AT03 through AT07.

Alternatively, you could create a server script for each audit database and then run multiple scripts when bringing up the system.

Audit databases are not specified for connection in the client scripts, as are the production, admin, and help databases. Instead, the connection is managed in MFG/PRO according to the setup data entered in Audit Database Maintenance (36.12.13.11). When audit information is written or when an audit report is run, the system automatically connects to the appropriate databases.

The audit databases can be included in your production database server scripts. This reduces the number of scripts to run, and the production database portion of the scripts can remain static. Alternately, you can create separate scripts that only launch the audit database servers. This is up to the administrators at each company.

Important You must have a new audit database available on the day you have assigned for the switchover in MFG/PRO. If a new audit database is not available, the audit information is written to a staging location in your production database, and this table can grow excessively.

Database Access Privileges for QADAUD

The Audit Trails infrastructure connects dynamically to the appropriate audit databases based on the dates that are used for processing, only when needed. This is true both when the creation process switches databases and when locating the correct databases for the date range of a report.

When the audit trail databases are connected through shared memory, users must have read and write access to the physical database files in the operating system. Otherwise, a Progress error message is generated regarding a shared memory access violation.

The fact that users must have write access to the database files may present a security issue for some companies. This situation can be avoided if the connection to the databases is client-server. Client-server connections can be used across machines or where both the client and the server are on the same machine.

The write access issue applies to the currently active audit database; this is the only one that the creation process updates. In Figure 3, this is database AT05. The databases that are no longer written to by the creation process—in Figure 3, databases AT02, AT03, and AT04—can have file access permission set to read-only for all users. The database is then connected in read-only mode (Progress startup option –RO). This can be defined in the startup (`.pf`) file for the respective audit database. The databases accessed with the –RO option should not have a database server started.

Storing Audit Trails Permanently

The process that stores the audit trails permanently is the second step in audit trail creation. It runs in the background and is launched as an MFG/PRO batch process. Its purpose is to convert any temporary audit trail data into the form of permanent audit trail records that can be reported against in the audit database.

After the permanent audit trail data is created, the corresponding temporary data is deleted. The audit trail creation process converts the temporary audit records from any audit-trailed table in a first-in, first-out manner and remains essentially idle if no temporary audit records need to be converted.

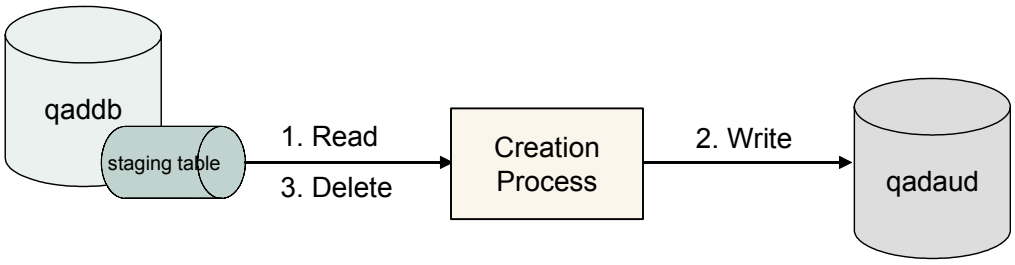
When data is written to two databases within the same transaction, it is possible that the changes to one database fail while the changes to the other commit. Progress provides two-phase commit functionality to deal with this problem. However, two-phase commit also requires after-imaging to be enabled for the two databases. The addition of two-phase commit and after-imaging causes a significant burden on the server that adversely affects system performance. The audit trail creation process has been coded to ensure the integrity of the permanent audit trail data so that the costly Progress two-phase commit and after imaging are not required.

The audit trail design restricts the processes that must span the two databases to the creation process alone. Normal MFG/PRO user activity does not require a connection to the audit database. This gives an additional level of security and some protection against unauthorized use of the audit database.

See "Two-Phase Commit" on page 11 for additional details.

The creation process is designed to allow multiple processes to be running simultaneously if necessary to keep up with the volume of records in the staging table. This feature provides for upward scalability.

Figure 4
Audit Creation Flow



Audit Trail Creation Process: Log File

The creation process has been designed run in the background—requiring no user interaction or monitoring. Significant activity in the process is reported to the log file. This file is named when the process is started.

If something goes wrong in the creation of the permanent audit trail records—for example, the process is unable to connect to the audit database—entries are created in the log file and an e-mail is sent to the system administrator group. If errors occur that pertain to a particular audit record, the record is saved to an error table in the database (aterr_mstr).

Records in the aterr_mstr table are reprocessed by the audit trail creation process after the error has been corrected.

Reporting from Audit Trails

When a user runs a report to see the audit trail for a record, the application connects to the appropriate audit databases that are available on disk. The session remains connected to the audit database while the report runs and then is disconnected when the report finishes. The databases are connected dynamically as needed, depending on which databases were active for the date range the user has selected for the report.

The report logic prints the history from all the databases that are available online when the report runs. If older databases have been archived, they can easily be restored and made available for reporting.

See *User Guide: MFG/PRO eB2.1 New Features* for further details on audit trail reporting.

Electronic Record to Audit Trail Linkage

To facilitate the linkage between MFG/PRO records and their audit trails, a unique record identifier is present for every record in the production database. This object ID (OID) field is populated by a create trigger that has been added to each table.

Using this OID allows the system to build generic links between the audit trail data and any record in the production database.

Adding an OID field and create trigger to each table is one of the steps necessary to enable audit trails for custom tables. This is described in “Customizations” on page 15.

Special Considerations

This section discusses additional topics that a database administrator should consider when the Enhanced Controls module is installed.

Making Schema Changes to Audited Tables

The temporary audit trail data is kept in a table named attmp_mstr. No reports access the data in this table. Therefore, QAD recommends that the creation process runs constantly to keep the audit trail database as up-to-date as possible for accurate audit trail reports.

Important This table must be empty before any changes are made to the schema of tables with audit trails turned on.

The programming technique used for the audit trail creation process requires that the schema definition of the production database table must be exactly the same as when the trigger fired. Therefore, make it a habit to run the audit trail creation process before any schema changes are applied.

Fields Deleted From a Table

A number of performance enhancements that affect Enhanced Controls functions are introduced in MFG/PRO eB2.1 SP3. As a result, beginning with SP3, the audit trail capture logic has an additional requirement. To be audited, a table must never have had a field removed since it was originally created.

When QAD delivers schema changes to existing tables, typically fields are not deleted. Instead they are renamed to QAD reserved fields. However, some exceptions to this approach have occurred in the past, and this rule may not have been followed for custom tables in your environment.

Once detected, this problem can be corrected. QAD provides a utility with MFG/UTIL, the standard installation tool for QAD products, that detects and corrects this problem. The correction involves removing and re-creating the table with the same table definitions and, of course, all its data intact.

This condition is detected and fixed when the Enhanced Controls module is installed. It is also validated each time a table is added to the list of tables to be audited. The utility in MFG/UTIL can be run to correct this situation even after the product has been in use for a while—not just when it is first installed.

System Configuration

This section discusses general hardware and software considerations for using Audit Trails. It also contains benchmark results produced by QAD.

Disk Configuration

With the addition of audit trails, significantly more data can be written to disk, depending on which tables are being tracked. This additional activity is spread across the database, before-image, and after-image files. QAD testing indicates as much as 2 to 3 times more data is written to both the database and the before-image file when all tables are audited.

As a general recommendation, QAD advises you to review the current volumes of data written to disk and determine the resources still available to handle the extra load. Specific recommendations are difficult to make because they would need to be tailored for each implementation. What would be a good idea in one case may not make sense in another.

After-Image Logging

Enabling audit trails includes no inherent requirement that after-image (AI) logging be enabled. Therefore, your approach to AI logging should not need to change because Enhanced Controls is installed. However, QAD recommends using AI logging in general for all online activity.

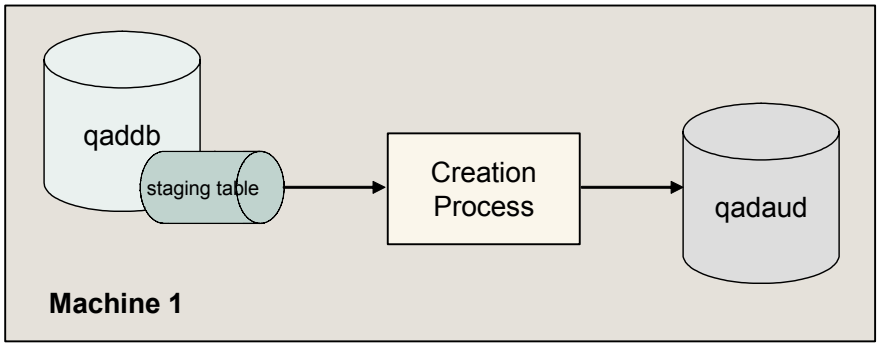
As mentioned previously, using Audit Trails can cause a significant increase in volumes written to disk, including the after-image file. This should be considered when reviewing the disk configuration as a part of the Audit Trails implementation.

For further details on AI logging, review the Progress deployment documentation.

CPU Configuration for Audit Trails

In the simplest implementation of Audit Trails, all the various components run on one machine, including the production and audit databases along with their database servers and the audit trail creation process. Running everything on one machine requires less overall administration of the environment.

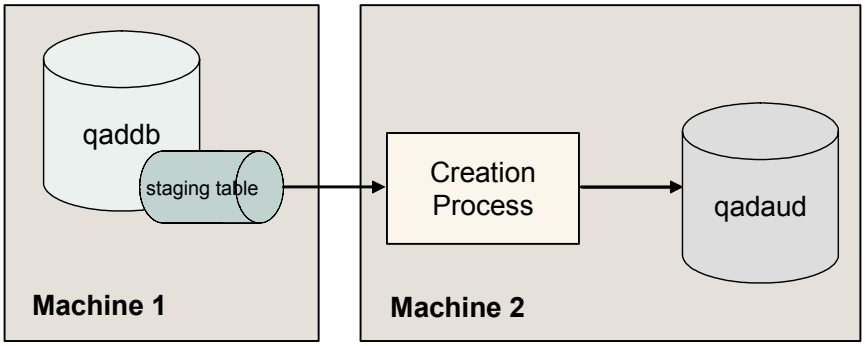
Figure 5
Audit Trails and QADDB on One Machine



However, when the volume of activity grows and one machine cannot handle all the processes, several low cost options exist to achieve scalability from the architecture. Since the audit database is separate from the production database, it can be easily moved to a different machine.

For most companies, the main process that uses the audit database is expected to be the creation process. It may be very active depending on the number of records captured by the audit trails. Therefore, moving it to a machine different from the main production machine as shown in Figure 6 will off-load that processing from the machine that primarily supports MFG/PRO.

Figure 6
Audit Trails and QADDB on Separate Machines



QAD has found that a reasonably fast Windows or Linux-based desktop machine can easily handle the creation process for even a large online system. A system like this allows more than one creation process to be started to increase its throughput.

When large batch jobs are running, the creation process could lag behind the batch job, but it will catch up when the batch job completes and audit trail activity decreases. The size of the lag and the time it takes to catch up will of course vary greatly with the specific conditions.

Running the audit database and the audit trail creation process on a separate machine from the main MFG/PRO production system forces a client-server connection between the creation process and QADDB. Since this connection is done from one machine to another, it will result in increased network traffic. QAD recommends that you monitor the impact this has on the network. If the network is flooded because of

this setup, you can easily remedy this by adding a separate network between these two machines that is dedicated only to the client-server connection to the Audit Trails server.

The usage of the audit database for reporting is expected to be low and contribute little to the load on that machine. Its load will come almost solely from the creation process.

Creation Process

The previous section discussed where to run the creation process. When you run the creation process on the same machine as the MFG/PRO run-time environment, you still have choices on how to run it. The creation process can connect to audit database and the production database using either shared memory or a client-server connection. Using shared memory for both databases gives significantly better performance.

When running the audit trail creation process and the audit database on a separate machine, you must use client-server to communicate between the creation process and the production data server. The audit trail creation process should connect to the audit database using shared memory for optimal performance.

Two-Phase Commit

In some cases when more than one database is involved in a single transaction, it is necessary to invoke Progress two-phase commit to ensure the integrity of the transaction across the two databases. With the current version of Progress (version 10.0 and earlier), using two-phase commit incurs a significant performance overhead. See the Progress documentation for details on two-phase commit.

The design of Enhanced Controls includes its own two-phase commit logic, eliminating the need to turn on two-phase commit when using Audit Trails and avoiding the associated performance penalty.

In some circumstances—such as when an MFG/PRO production database has been split or customized tables exist in a separate database—the site may have decided to turn on two-phase commit in order to ensure the integrity of the system.

QAD, however, has found a significant overhead associated with the two-phase commit. For some operations, the overhead has resulted in as much as 5 to 50 times longer processing time. Therefore, QAD recommends keeping all production tables in the one QADDB database—including custom tables—to avoid the need for two-phase commit.

Expected Performance

Adding the overhead of auditing will result in higher demands on your hardware systems. Performance will vary based on the tables you choose to audit, the volume of records generated for those tables, and how you structure your file system. You should run tests on a non-production server that simulate your business model and anticipated audit needs so that you can understand the impact on your system.

It is possible that you will need to reconfigure or obtain additional hardware resources to bring your performance back in line with your standards. You should carefully choose which tables need to be tracked; tracking changes to tables you do not need to report against will unnecessarily degrade the performance of the system.

QAD has performed several benchmarks to determine the impact of turning Audit Trails on for all tables in the MFG/PRO system. The benchmarks were extreme since it is unlikely that any customers will enable Audit Trails for all tables. The results of these extreme benchmarks show a non-noticeable impact on UI activities. A noticeable impact was observed on long-running batch processes such as Materials Requirement Planning (MRP). The next section describes the numbers that have been recorded in the benchmark testing.

Note As discussed previously, it is normal and expected for auditing to cause some performance degradation unless hardware resources are increased to compensate.

QAD can help you determine if Audit Trails will adversely affect the performance of your production MFG/PRO system by performing a Q/Scan technical services evaluation of your system. Contact your QAD account representative for more information.

Benchmark Data

QAD has run benchmarks to determine the impact on a system with auditing enabled compared to the same system without audit trails. These tests have been run on a pristine test machine. The disk configuration for storage areas and before-image (BI) files was optimized for the scenarios tested.

For the UI benchmarks, the machine used was:

- Machine: Dell PowerEdge 2600
- CPU: Dual 2.6 GHz Xeon, 4 GB RAM

For the MRP benchmarks, the machine was:

- Machine: Dell 1600
- CPU: Dual 2.4 GHz Xeon, 1 GB RAM, 533 MHz FSB
- Disks: 18 GB 10Krpm SCSI disks, 4 disks for the database, 2 disks for BI files, 1 disk for O/S

Tests were made with a subset of transactions and 200 simulated users to simulate normal UI activity. The transactions used were Purchase Order Maintenance, Inventory Unplanned Issue, Inventory Unplanned Receipt, Sales Order Inquiry, and Customer Inquiry. Table 1 lists the results.

Table 1 Cross-Section of UI Programs Tested on eB2.1 SP2[1]

Activity	Relative Time
Audit Trails not installed	100%
Audit Trails installed and turned on for all tables; no creation process running	+12%

1. This version of MFG/PRO eB2.1 included a set of related ECOs available on the QAD Support Web site.

Another test was set up to determine the impact of audit trails on long-running batch jobs such as MRP. The MRP test was executed on a copy of a large production database. Table 2 lists the results.

Table 2 MRP Test Results for MFG/PRO eB2.1 SP2[1]

Activity	Relative Time
Audit Trails not installed	100%
Audit Trails installed and turned on for all tables No creation process running	+136%

1. This version of MFG/PRO eB2.1 SP2 included a set of related ECOs available on the QAD Support Web site.

More complete test results for the impact of audit trails on MRP were executed on MFG/PRO eB2.1 with Service Pack 3. Table 3 lists these results.

Table 3 MRP Test Results for eB2.1 SP3

Activity	Relative Time
Audit Trails not installed	100%
Audit Trails installed and turned on for all tables No creation process running	+67%
Audit Trails installed and turned on for all tables QADAUD and creation process running on same machine as QADDB	+135%[1]
Audit Trails installed and turned on for all tables QADAUD and creation process running on different machine from QADDB	+90%[1]
Audit Trails installed and turned on for relevant tables: wo_mstr, wod_det, wr_route Audit trails turned off for in_mstr, mrp_det, oa_det, pfc_det, pk_det, qad_wkfl No creation process running	+45%

1. This was the result for the MRP run. The creation process was still running when the MRP process finished.

Planning an Audit Trails Implementation

This section describes some of the topics you should consider before enabling Audit Trails for an MFG/PRO system.

*User Guide: MFG/PRO eB2.1 New Feature*s includes complete details on enabling auditing for selected tables.

Selecting Tables for Audit Trails

You must determine which tables you want to audit. This is a significant analysis task that should not be underestimated. The task is important for two reasons:

- You must determine what tables need to be tracked to support compliance with regulatory requirements such as CFR Part 11, Sarbanes-Oxley, IFRS, or other internal control requirements.
- You must balance the amount of data collected against the impact on the system resources such as disk space and response times.

Audit Trails functions let you selectively identify each table you want to audit.

Tables in MFG/PRO can be categorized as follows with common characteristics to consider regarding audit trails:

Master tables. These can be crucial to the integrity of your business processes and are maintained from the UI by end users. Examples of these tables are engineering data such as items, routings, and bills of material.

Transaction tables. These are used to track supply, demand, and financial data and are also maintained from the UI by end users. Examples include tables such as purchase orders, sales orders, work orders, AP vouchers, and their related detail tables.

System transaction tables. These, in general, are not maintained directly by users, but are updated by the system as a side effect of user actions. Examples are MRP detail, inventory transaction history, operation history, and invoice history. System validations are intended to ensure that these records are accurate and complete as a result of normal system activity. The history tables were originally intended to provide an audit of key system activity, although they do not include some of the details required for strict audit trail compliance, such as full user name.

Each category of table involves different levels of risk depending on the purpose of the audit trails and the compliance issue being addressed. Many companies will take a risk-based approach when deciding which tables to audit.

Installation Considerations for eB2.1

In order to use audit trails, you must first activate the OID generation facilities. This requires minor schema changes and the initial generation of OIDs for all existing records. A utility is provided for this activity. The utility both turns on the OID generator for all future transactions and generates OID values for all existing database records. This is required in order for the Audit Trails functionality to work.

The time it takes to complete this activity depends on the number of existing records and the capability of your hardware systems. Before beginning, you should run some tests to get a good indication of how long this process will take.

Customizations

The design for audit trails has been carefully crafted so that the functionality can be easily extended to any tables in the database, including custom tables. The procedure for audit trail enabling tables has been described in the QAD development standards. These are available to users with a QAD Web account on the QAD Support page:

http://support.qad.com/

The following standards are particularly applicable:

- DST-0005: Developing with OID fields
- STD-0326: Every qaddb table must have a create trigger
- STD-0331: Every qaddb table must be audit trail enabled

These standards describe two main points that need special attention. One is the schema definitions for tables that are audit trail enabled. The second is what code constructs need to be modified to avoid run-time errors from custom code that modifies MFG/PRO data.

The standards listed here describe what needs to be considered. QAD Consulting can also assist in converting custom code to support audit trails.

Appendix A

Audit Database Sizing

This appendix includes reference information related to MFG/PRO audit databases required by the Enhanced Controls module.

Introduction

An audit database can be configured to store only audit trail data, only electronic signature archive data, or both. Guidelines are provided in this section for setting up storage areas and extents for audit databases. However, audit databases will vary greatly in size depending on their configured use, the volume of changes being audited, and the volume of electronic signature data being archived. In addition, the database sizing guidelines in this section are based on data from a test environment and may not apply to your company's data. Live database configurations should be determined based on site-specific analyses.

When configuring a Progress database, the following considerations must be made and reflected in the data definition file (.df) and structure description file (.st) for the database:

- Block Size: Each Progress database is created with a single block size of 1, 2, 4, or 8K. This means that all memory blocks in the database are the same size. By default, MFG/PRO databases are created with an 8K block size.
- Storage Area Assignments: Storage areas give you physical control over the location of specific database objects. Storage areas can contain any combination of tables and indexes, but you cannot split a table or index across storage areas. Each table and index may be assigned to only one storage area.
- Records per Block: Each storage area in the database has a specific number of records per block. Storage areas can have 1, 2, 4, 8, 16, 32, 64, 128, or 256 records per block. The Progress default number of records per block is 64 if the block size is 8K, and 32 for all other block sizes. The records per block value can be chosen to maximize storage to disk (and minimize empty storage) by taking into account the mean record size to be stored in the storage area.
- Extents: An extent corresponds to a single OS file of a fixed or variable size. Each storage area can be split into one or more extents that allow you to extend storage areas across multiple files on the same or different physical volumes. The number and size of these extents must accommodate the data that will be stored during the lifetime of the database.

Audit Database Default Values

By default, all MFG/PRO databases are created with an 8K block size.

Based on actual audit trail data from a test environment, QAD has provided default storage area assignments and records per block values for those storage areas. These default values should work well for most companies, but if your company has chosen a different block size, then the default records per block values will need to be adjusted. The calculations of these default values are described in sections that follow.

The default structure file shipped for the audit database was modified somewhat from these results to balance the efficient storage of data with the size of the storage area. Specifically, any calculated record per block value below 64 was changed to 64 to keep the maximum size of the storage area to a manageable 256 GB.

The default extent sizes provided by QAD are not intended to meet the audit database storage needs of all companies and must be adjusted for each installation based on analysis of actual data. The default configuration provided for the audit database requires a total of 4GB for the fixed extents and is intended to be used only in a trial situation to determine the actual extents needed. This process is described in “Extents” on page 24.

Progress Tools

Database Record Analysis

The Progress `dbanalys` tool that is a part of the proutil utility can be used to analyze existing data in a database. As discussed in the sections that follow, this data is useful for determining the records per block and extent sizes. Run dbanalys from the Progress install `\bin` directory as follows:

```
proutil \mfgsvr\db\audprod.db -C dbanalys > out.txt
```

Go to the Record Block Summary portion of the report to view a breakdown of each storage area in the selected database. This summary shows the number of records for each table in the storage area, as well as the size in bytes, and the minimum, maximum, and mean record size in bytes. The Index Block Summary follows the record summary and shows similar information by index.

Storage Area Utilization

To monitor the free space in storage areas, the following sample 4GL code can be executed from a Progress Editor session connected to the database of interest:

```
for each _areastatus no-lock where _areastatus-areanum > 5:
    display
        _areastatus-areaname format "x(14)" label "Area"
        _areastatus-totblocks format ">>>,>>>,>>>" label "Total"
        _areastatus-hiwater format ">>>,>>>,>>>" label "Hi Water"
        _areastatus-totblocks - _areastatus-hiwater
            format ">>>,>>>,>>>" label "Free"
        _areastatus-hiwater / _areastatus-totblocks * 100
            format ">>9%" label "%Full".
end.
```

For an audit database, this will produce a display similar to the following:

```
Area                  Total    Hi Water         Free %Full
--------------- ----------- ----------- ----------- -----
Schema Area             207         203           4   98%
ATMSTR               32,002      15,702      16,300   49%
ATMSTR_IDX1          16,002       2406      13,596   15%
ATMSTR_IDX2          16,002       1011      14,991    6%
ATMSTR_IDX3          16,002       2605      13,397   16%
ATFDET              250,002     113,008     136,994   45%
ATFDET_IDX1          16,002       2402      13,600   15%
ATFDET_IDX2          16,002       2709      13,293   17%
ATKDET               32,002       5,407      26,595   17%
ATKDET_IDX1          16,002       1304      14,698    8%
ATKDET_IDX2          16,002        513      15,489    3%
ATKDET_IDX3          16,002       2603      13,399   16%
```

```
AESIG                32,002           3       31,331     0%
AESIG_IDX            32,002          14       31,988     0%
```

MFG/PRO Database

Although the topic of this chapter is the audit database, this section describes the database configuration changes to one of the standard MFG/PRO databases. New storage areas have been designated in the `mfgprod` database for Enhanced Controls. These storage areas are not intended for long-term storage of this data, and so the sizing considerations are somewhat different than for the audit database.

Audit Trails

A new storage area, ATTMPMSTR, has been designated in the `mfgprod` database for the staging table that receives audit trail data during production activities. This data is then flushed to the audit database and deleted from the attmp_mstr staging table by one or more Audit Trail Creation Processes running in the background. The indexes for the staging table have been assigned to a new ATTMPMSTR_IDX storage area. Sizing for these storage areas should be of minor concern. The storage requirements will vary depending on the number of audited tables, frequency of changes to those tables, and number of background processes running. In general, you can control the required size of these storage areas by managing the number of background processes launched.

Start with one process per fifty users and monitor the table size throughout normal production activities. If you observe steadily increasing table size, increase the number of background processes accordingly.

Important If the background processes are shut down for some reason or if the audit database is not online, records remain in the staging table and could overrun the storage area assigned. Be sure to monitor the free space in these storage areas on a regular basis as described in “Progress Tools” on page 19.

The new storage area definitions in the default `mfgprod.st` structure file do not specify fixed extents and only specify one variable extent for each as shown below:

```
#
d "ATTMPMSTR":39,64 ./db
#
d "ATTMPMSTR_IDX":40,32 ./db
#
```

Note The structure file syntax above shows a comment sign (#) followed by the definition of one 64-record-per-block variable extent for the ATTMPMSTR storage area and is located in the `db` subdirectory. A variable extent can reach a maximum of 2GB in size.

See “Extents” on page 24 for additional information on sizing extents.

These storage area definitions will need to be modified if audit trails will be enabled. Add one or more fixed extents to be consistent with the considerations described above and your sizing strategy for this database.

The sample below has been modified to provide one fixed extent (f) with a size of 5MB (5120) in addition to the variable extent for each storage area:

```
#
d "ATTMPMSTR":39,64 ./db f 5120
d "ATTMPMSTR":39,64 ./db
#
d "ATTMPMSTR_IDX":40,32 ./db f 5120
d "ATTMPMSTR_IDX":40,32 ./db
#
```

Electronic Signatures

A new ESIG storage area has been designated for the electronic signature transaction tables that have been added to the `mfgprod` database. The corresponding indexes have been assigned to a new ESIG_IDX storage area. Sizing for these storage areas should be based on your anticipated use of electronic signatures. The storage requirements will vary depending on the number of electronic signature-enabled programs, level of activity on those programs, and your archive/delete schedule for electronic signature data.

E-Signature Archive/Delete (36.12.14.22) can be used to archive electronic signature data to an audit database. Unlike for audit trails, the data is not automatically transferred to the audit database by a background process. During the archive/delete process, not all signature data will be deleted even if the data meets the specified date range. The latest signature for any signed record in MFG/PRO will not be deleted because that signature information must be available for display whenever the record is displayed or modified.

Important Be sure to monitor the free space in these storage areas on a regular basis as described in “Progress Tools” on page 19 and run the archive/delete menu program as needed.

The new storage area definitions in the default `mfgprod.st` structure file do not specify fixed extents and only specify one variable extent for each as shown below:

```
#
d "ESIG":41,64 ./db
#
d "ESIG_IDX":42,32 ./db
#
```

These storage area definitions will need to be modified if electronic signatures will be enabled. Add one or more fixed extents to be consistent with the considerations described above and your sizing strategy for this database.The sample below has been modified to provide one fixed extent (f) with a size of 5MB (5120) in addition to the variable extent for each storage area:

See “Extents” on page 24 for additional information on sizing extents.

```
#
d "ESIG":41,64 ./db f 5120
d "ESIG":41,64 ./db
#
d "ESIG_IDX":42,32 ./db f 5120
d "ESIG_IDX":42,32 ./db
#
```

The remainder of this section focuses on the sizing considerations of the audit database.

Storage Area Assignments

See the Progress *Database Administration Guide and Reference* for detailed background on storage areas.

Audit Database

The audit database `audprod` contains three audit trail-related tables—atf_det, atk_det, and at_mstr—and eight indexes. By default, each of these tables and indexes is partitioned into its own new storage area for maximum storage capacity and flexibility.

The database also includes four electronic signature archive tables—aescd_det, aescx_det, aesig_mstr, and aesrec_det—along with ten indexes. Because the electronic signature volume will be relatively low, these four tables are assigned to one storage area and the corresponding indexes to a second storage area.

Table 4 shows the default assignment of each table and index to a storage area.

Table 4
Audit Database Storage Areas

Object Type	Used For	Object	Storage Area
Table	Audit Trail	atf_det	ATFDET
Index	Audit Trail	oid_atf_det	ATFDET_IDX1
Index	Audit Trail	atf_oid_at_mstr	ATFDET_IDX2
Table	Audit Trail	atk_det	ATKDET
Index	Audit Trail	oid_atk_det	ATKDET_IDX1
Index	Audit Trail	atk_field_name	ATKDET_IDX2
Index	Audit Trail	atk_oid_at_mstr	ATKDET_IDX3
Table	Audit Trail	at_mstr	ATMSTR
Index	Audit Trail	oid_at_mstr	ATMSTR_IDX1
Index	Audit Trail	at_table_name	ATMSTR_IDX2
Index	Audit Trail	at_oid_erecord	ATMSTR_IDX3
Table	E-Sig Archive	aescd_det	AESIG
Index	E-Sig Archive	aescd_oid_escat_mstr	AESIG_IDX
Index	E-Sig Archive	oid_aescd_det	AESIG_IDX
Table	E-Sig Archive	aescx_det	AESIG
Index	E-Sig Archive	aescx_oid_det	AESIG_IDX
Index	E-Sig Archive	oid_aescx_det	AESIG_IDX
Table	E-Sig Archive	aesig_mstr	AESIG
Index	E-Sig Archive	aesig_date	AESIG_IDX
Index	E-Sig Archive	aesig_userid	AESIG_IDX
Index	E-Sig Archive	oid_aesig_mstr	AESIG_IDX

Object Type	Used For	Object	Storage Area
Table	E-Sig Archive	aesrec_det	AESIG
Index	E-Sig Archive	aesrec_table_name	AESIG_IDX
Index	E-Sig Archive	aesrec_record_order	AESIG_IDX
Index	E-Sig Archive	oid_aesrec_det	AESIG_IDX

Records per Block

General Recommendations

The default records per block values for the audit database storage areas were determined using the following analysis of actual audit database data from a test environment. These defaults should work well for most companies because the mean record size per table is not expected to vary much from the test data analyzed.

Note This information assumes the default 8K database block size. If you are using a different block size, you will need to adjust the records per block values in the `audprod.st` structure file. You can substitute other database block sizes in the formulas provided.

To configure storage areas to maximize storage to disk (and minimize empty storage), you need to know the mean record size you are writing to disk. A simple formula for estimating the records per block is:

database block size / [mean record size + 20 (estimated row overhead)] = records per block

For example, if you are writing 1K records (mean size) to an 8K block size database (8192 bytes), you will want to create the storage area with 8 records per block.

8192 / (1024 + 20) = 8

When multiple tables are stored in the same storage area, the mean record size will need to be averaged over all tables as you will see in the specific examples. You will also not usually end up with a round number. Round the number up or down to the closest permissible records per block.

Audit Database

The audit database from a QAD test environment was analyzed using the proutil command and the output is summarized in Table 5.

Table 5 Record Size in Bytes for the Audit Database

Table	#Records	Size	Record Size		
			Min	Max	Mean
atf_det	134474	113.0M	63	3073	881
atk_det	82497	5.4M	54	94	69
at_mstr	134434	15.7M	96	150	122

Table	#Records	Size	Record Size		
			Min	Max	Mean
aescd_det	366	38.8K	97	112	108
aescx_det	559	559.4K	90	1506	1024
aesig_mstr	259	22.8K	75	119	90
aesrec_det	533	49.3K	82	99	94

This data was then used to calculate the default records per block for the `audprod.st` structure definition file. The calculation for the AESIG storage area is described here.

The total mean record size for a storage area is determined by adding 20 to each table's mean record size and averaging the means weighted by the number of records. For the AESIG storage area, the mean record size would be:

[128 (366) + 1044 (559) + 110 (259) + 114 (533)] / (366 + 559 + 259 + 533) = 419

The database block size (in bytes) is then divided by the total mean record size:

8192 / 419 = 19.6

This result is then rounded up or down to the nearest permissible records per block, which in this case is 16.

For storage areas that will store records for only one table, the total mean record size for the storage area will just be the mean record size for the table plus 20.

Table 6 shows the resulting records per block for the audit database.

Table 6 Number of Records per Block for the Audit Database

Storage Area	Database Block Size/ Mean Record Size	Recommended Records/Block
ATFDET	8192 / 901 = 9.1	8
ATKDET	8192 / 89 = 92	128
ATMSTR	8192 / 142 = 58	64
AESIG	8192 / 419 = 19.6	16

The default records per block for storage areas containing only indexes is 32 for all MFG/PRO databases. This default should work well for the audit database also.

Extents

General Recommendations

In general, fewer extents are optimal for improved performance. The fewer the extents, the less system resources are needed (such as file handles), the less time it takes to open the database, and the easier it is to administer the database.

The default recommendation for each extent size is just under 2 GB (2000000 in the structure file). However, a safe extent size of 0.5 GB is also acceptable for most large databases (512000 in the structure file). If the database is smaller than 2 GB, use a smaller fixed extent with one variable extent.

For databases where performance is not critical, use one or a few large fixed extents. Always have at least one variable extent as a safeguard.

You should focus on achieving good input/output (I/O) performance. To do this, the database should be spread out across multiple disk drives. The more drives, the greater the overall system I/O capacity. Progress recommends that you use striped disk arrays and mirrored disks, and that you isolate the before-image and after-image files.

See the Progress documentation for more information.

There is a maximum size for a storage area that is independent of the number of extents but dependent on the block size and records per block as shown in Table 7 for an 8K block size.

See the Progress documentation for the limits for other block sizes.

Table 7
Progress Maximum Storage Area Sizes

Database Block Size	Records Per Block	Maximum Area Size
8192 bytes (8K)	1	16TB
8192 bytes (8K)	2	8TB
8192 bytes (8K)	4	4TB
8192 bytes (8K)	8	2TB
8192 bytes (8K)	16	1TB
8192 bytes (8K)	32	512GB
8192 bytes (8K)	64 (default)	256GB
8192 bytes (8K)	128	128GB
8192 bytes (8K)	256	64GB

The default structure file shipped for the audit database was modified somewhat from these results to balance the efficient storage of data with the size of the storage area. Specifically, any calculated record per block value below 64 was changed to 64 to keep the maximum size of the storage area to a manageable 256GB.

Audit Database

The best way to determine the size of the storage area extents is to actually audit trail a live system for a short period of time and then analyze the resulting audit database. Alternatively, this can be done in a test environment as long as the activity of the live system can be simulated in the test environment. Create an audit database to use for the trial period using the provided default structure file, `audprod.st`. This database will require a total of 4 GB for the fixed extents. After determining the tables of interest to be audit trailed, configure audit trail profiles for those tables to begin on a particular day (Audit Trail = On). Audit trails will be generated starting at 12:01 AM. that day. Make sure to have an Audit Trail Creation Process running so that the audit trails will be transferred to the audit database. After running the system for the desired period of time, perhaps a few hours, one day, or even a few days, run the Progress dbanalys utility against the audit database, saving the output to a file.

Table 8 Sample Table Results of Database Analysis Report

Table	#Records	Size	Record Size		
			Min	Max	Mean
atf_det	1,075,792	904.0M	63	3073	881
atk_det	659,976	43.2M	54	94	69
at_mstr	1,075,472	125.6M	96	150	122
aescd_det	2,928	310.4K	97	112	108
aescx_det	4,472	4.475M	90	1506	1024
aesig_mstr	2,072	182.4K	75	119	90
aesrec_det	4,264	394.4K	82	99	94

Assuming that the numbers in Table 8 are from running audit trails in a live system for one day, you can estimate the required size for each storage area according to the life expectancy for the audit database.

Example If the audit database is to be active for 3 months (65 working days), the required storage area sizes would be calculated as shown in Table 9.

Table 9 Extent Sizing Based on Sample Analysis Report

Storage Area	Tables Included	1-Day Size	65-Day Calculated Size
ATFDET	atf_det	904.0M	58.76G
ATKDET	atk_det	43.2M	2.808G
ATMSTR	at_mstr	125.6M	8.164G
AESIG	aescd_det, aescx_det, aesig_mstr, aesrec_det	5.362M	348.5M

It is always possible and usually desirable to create multiple fixed extents to satisfy the calculated size of a single storage area. In fact, this would be required if the calculated size exceeds the maximum file limit of the operating system. Alternatively, the life expectancy of the database can be reduced. For example, a new database can be brought online every two months instead of every three months. If the calculated size of a storage area exceeds the Progress maximum size as listed in Table 7 then the life expectancy of the database must be reduced.

If a fixed extent is to have a size of 2 GB or larger, be sure to enable large files for the database using the following command:

```
proutil <db_name> -C EnableLargeFiles
```

The operating system also needs to have large files enabled. Refer to your operating system documentation for further information.

As a safeguard, always have at least one variable extent in addition to one or more fixed extents for each storage area.

Note Because a variable extent can grow to 2 GB if supported and enabled in the operating system, it is a good idea to enable large files in the database even if the fixed extents are less than 2 GB.

A similar analysis of the index sizes can be used to determine the extent sizes for those storage areas as described below.

Table 10 Sample Index Results of Database Analysis Report

Table	Index	Size
atf_det	atf_oid_at_mstr	21.6M
	oid_atf_det	19.2M
atk_det	atk_field_name	4.08M
	atk_oid_at_mstr	20.8M
	oid_atk_det	10.4M
at_mstr	at_oid_erecord	20.8M
	at_table_name	8.00M
	oid_at_mstr	19.2M
aescd_det	aescd_oid_escat_mstr	48.8K
	oid_aescd_det	43.2K
aescx_det	aescx_oid_aescd_det	64.0K
	oid_aescx_det	57.6K
aesig_mstr	aesig_date	21.6K
	aesig_userid	23.2K
	oid_aesig_mstr	31.2K
aesrec_det	aesrec_record_order	60.8K
	aesrec_table_name	36.0K
	oid_aesrec_det	62.4K

Assuming that the numbers in Table 10 are the dbanalys output for the one day of system use, the required extents would be calculated as shown in Table 11.

Table 11 Index Extent Sizing Based on Sample Analysis Report

Storage Area	Indexes Included	1-Day Size	65-Day Calculated Size
ATFDET_IDX1	oid_atf_det	19.2M	1.25G
ATFDET_IDX2	atf_oid_at_mstr	21.6M	1.40G
ATKDET_IDX1	oid_atk_det	10.4M	676.M
ATKDET_IDX2	atk_field_name	4.08M	265M
ATKDET_IDX3	atk_oid_at_mstr	20.8M	1.35G
ATMSTR_IDX1	oid_at_mstr	19.2M	1.25G
ATMSTR_IDX2	at_table_name	8.00M	520M

Storage Area	Indexes Included	1-Day Size	65-Day Calculated Size
ATMSTR_IDX3	at_oid_erecord	20.8M	1.35G
AESIG_IDX	aescd_oid_escat_mstr, oid_aescd_det, aescx_oid_det, oid_aescx_det, aesig_date, aesig_userid, oid_aesig_mstr, aesrec_table_name, aesrec_record_order, oid_aesrec_det	449K	29.2M