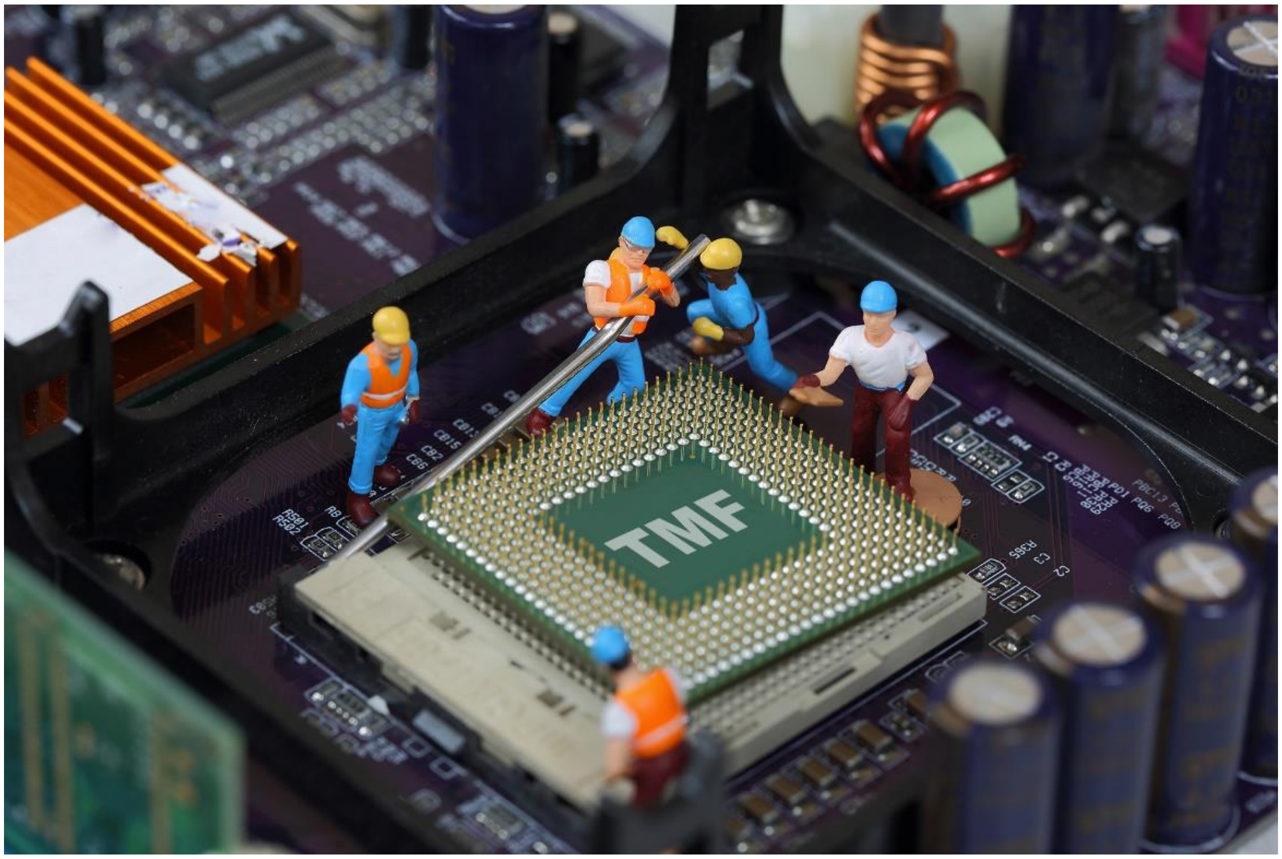
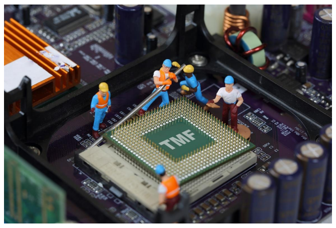
Only the Truth:
Debunking HPE NonStop TMF Data Protection Myths

A Gravic, Inc. White Paper



Note: this work was originally published as [an article in the Connection]()

Executive Summary



We feel it is critically important to address this topic because considerable misinformation still exists regarding the Transaction Monitoring Facility (TMF) and data auditing in general on HPE NonStop servers. Perceptions remain that the use of TMF is superfluous, costly, or has bad performance issues, yet *nothing could be farther from the truth!*

These perceptions, along with a lack of understanding or disregard for the data integrity protection benefits offered by TMF, can dramatically and negatively affect your organization – *ignore this information at your own peril.*

Debunking these myths and leveraging this transformative technology is critical to empowering mission- and business-critical applications and services, and for preserving along with competitively positioning NonStop applications well into the future.

In fact, far from being an impediment, having a TMF-audited database on your NonStop systems offers significant operational, reliability, and data integrity advantages, and also improves overall system performance, reduces application response times, and can even lead to improved capacity utilization along with application scalability.

Switching from a non-audited to an audited application environment can be daunting and quite frankly impossible if you license the application from another entity or otherwise cannot access the application source code. However, it is simplified when using the facilities provided by HPE AutoTMF as these avoid application modifications, and do not introduce significant overhead.

The myths and other objections raised against the use of TMF auditing and AutoTMF simply do not stand up to scrutiny. Using TMF (or AutoTMF):

- Ensures database consistency and brings an unaudited database up-to-date to the best practices available for DBMS data management.
  - Therefore, it may help to save one's application (or even job) and database from being replaced by the technology du jour that management perceives as *more modern*.
- Helps to future-proof applications, enabling the exploitation of new capabilities that are based on TMF usage (such as zero data loss).
- Dramatically improves, rather than hinders, application and database performance.
- Provides a lower Total Cost of Ownership (TCO) than the costs incurred from inconsistent or missing data (yes, this *still* happens in the twenty-first century IT world).
- Avoids the need to modify the application.
- Enables different business continuity and disaster recovery options. These options include architectures that can use the TMF recovery process by leveraging TMF dumps, or use data replication to create a consistent backup database that is complete and correct. This introduces significant architectural benefits over non-audited application environments.

Thus, the path to improving your overall non-audited environment requires a TMF implementation. Audited environments can even be implemented alongside existing non-unaudited applications.

As the creators and developers of HPE Shadowbase solutions, we feel strongly about helping companies improve their NonStop applications and databases by implementing audited architectures. We will gladly assist with running a trial or Proof-of-Concept (PoC) alongside our HPE partners and colleagues, using your application and your data, in your IT environment.

Seeing is believing!

Table of Contents

Table of Figures

Only the Truth: Debunking HPE NonStop TMF Data Protection Myths

It's All About Data Integrity and Data Availability

It all starts with protecting your valuable data. Not in the *data security* sense (meaning encryption or tokenization), although that certainly is important and receives much press. Rather, this paper focuses on several data integrity aspects, making sure your data is actually correct, consistent, and complete, as well as protected with the proper tools: TMF and data replication. Although data security is important, please refer to the HPE website for additional information on that topic.

The Transaction Log

HPE Shadowbase data replication software captures change data (data created or updated by users and applications) from a transaction log. As data is changed, the transaction manager writes the changes to the transaction log, where they are read by HPE Shadowbase software for replication to a target system or application.

On an HPE NonStop system, this transaction log is maintained by the HPE NonStop Transaction Management Facility (TMF) subsystem, and is known as the TMF Audit Trail (TMF AT). Data managed in this way by TMF are known as audited files, or audited data (conversely, non-audited-files and non-audited data)[1]. As changes are made to audited data (Enscribe files, SQL/MP and SQL/MX tables), these changes are permanently made to the TMF AT when the application's transaction commits, or is rolled back (undone) as the application's transaction aborts. The changes are then read and replicated by HPE Shadowbase data replication. Therefore, for the replication engine to operate, the changes to data must be TMF audited.[2]

Myths from the Early Days

However, in the early days of Tandem Computers, this TMF auditing function was not particularly efficient, and could negatively impact application performance. These issues were quickly resolved, but the reputational damage to TMF was already done.

A myth evolved that using TMF audited files and tables should be avoided at all costs if you wanted your application to perform well. This myth took root, and is unfortunately still prevalent today, even though times have significantly changed, and *the myth is now manifestly untrue*.

As a consequence of this myth and other falsehoods, we sometimes meet resistance from customers when they learn that for HPE Shadowbase replication to function, they must use audited files and tables. *We have found that the two main myths cited in opposition are:*

1. *TMF will slow down my application.*
2. *Using TMF will require an application rewrite.*

These old myths and any other newly proposed reasoning against TMF for performance are simply false. In fact, TMF protection and HPE Shadowbase software strategically integrate to provide the highest HPE NonStop application availability and data integrity to date.

Debunking the Myths

MYTH: *TMF Will Slow Down my Application*

TRUTH: Far from creating performance issues, the use of TMF auditing will almost always result in *dramatically* improved application performance. A primary reason for this myth is because it evolved during the period before significant enhancements were made to TMF and the disk process (DP2) software.[3]

[1]Note: In this paper, non-audited files and data includes non-audited SQL tables and is interchangeable with the term "unaudited files and data."
[2]There are a few exceptions to this rule, but they are not germane to this discussion.
[3]DP2 is an HPE NonStop software component that provides the interface between TMF, the database, and file subsystems.

These enhancements include:

- Most importantly, the database disk processes can eliminate physical I/O operations by caching if updates are audited[4] with no possibility of data loss. When a system fails, updates in the audit trail are reapplied instead of being lost.
- TMF overhead is minimal, since it is implemented as part of the NonStop OS kernel and disk process. HPE NonStop as a platform was optimized for transaction processing; therefore, transaction processing requires a low-level of resources.
- Audited records sent from the database disk process to the audit trail disk process are blocked together, using a technique called *boxcarring*. A few shorter messages can efficiently represent a large number of transactions.
- Audit trail writes are also boxcarred. Audit for many transactions from many disks is collected and written to the end of the audit trail with a single I/O.
- TMF itself can be massively parallelized using many parallel audit trails (i.e., master and auxiliary audit trails).

These enhancements have busted the myth about poor performance when using audited data. Since some may still be skeptical, a performance analysis was undertaken[5], comparing the transaction rates and response times for a sample application using audited and non-audited files. Much of the following performance discussion was taken from that study.

In this analysis in Figure 1, the absolute best transaction rate that could be achieved using non-audited files was 960 transactions per second (TPS), whereas using audited files reached 2,450 TPS. Note that this best non-audited transaction rate was achieved when using database buffering, which comes at the expense of possible data loss in the event of a failure – something which is not possible when using audited files.[6] For the same risk of data loss between file buffering vs TMF protection (using an unaudited, unbuffered database), only 150 TPS was achieved, far below the rate that the audited files achieved.

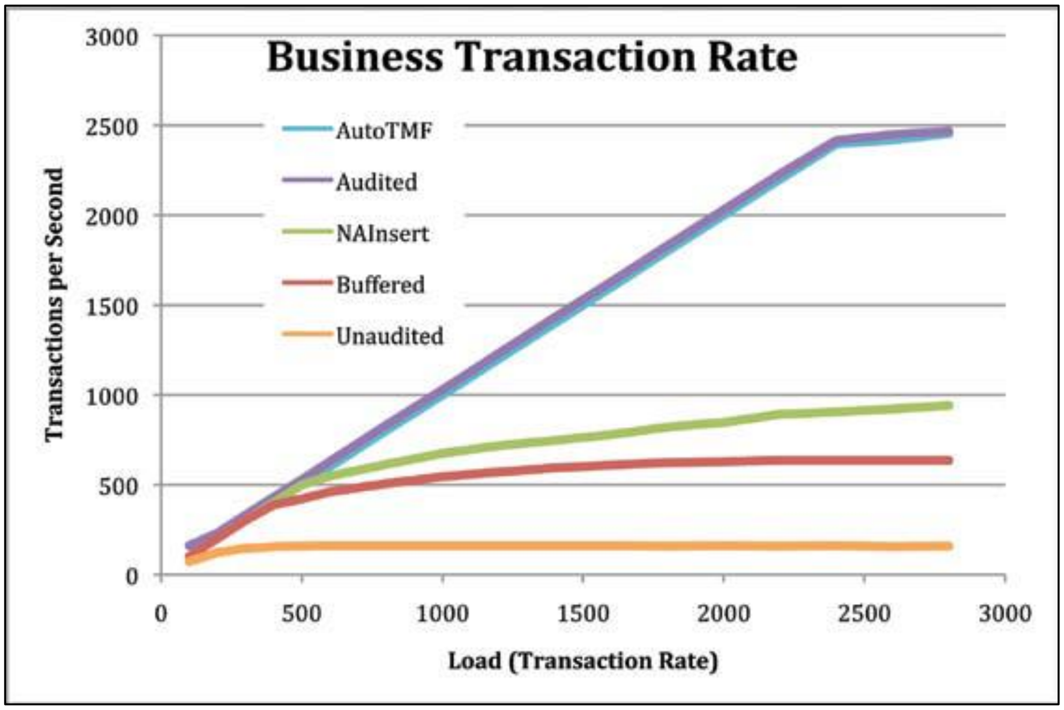


Figure 1 — Audited vs. Non-Audited Transaction Rate

[4]To achieve optimal performance, ensure that the DP2 disk cache size is set to a sufficiently large value. As a rough guide, the DP2 disk cache should be at least 10-20% of the total amount of main memory available. DP2 disk cache size can be checked for each data volume by using the SCF INFO <volume>, CACHE and SCF STATS <volume> commands.

[5]*Best Practices: Using TMF to Implement Business Continuity/Disaster Recovery*, Richard Carr, Carr Scott Software Inc., *The Connection*, Sept-Oct 2013 | Volume 34, No. 5.

[6]Note that "file buffering" (setting the BUFFERED file attribute) is different than TMF data caching. BUFFERED data can be lost if a failure occurs. However, TMF-cached data is always guaranteed to be recoverable.

The same result is true for transaction response times in Figure 2. For the same load on the system, when using unaudited files, the transaction response time was as much as 10x slower (> 500 ms) than when using audited files (< 50 ms).

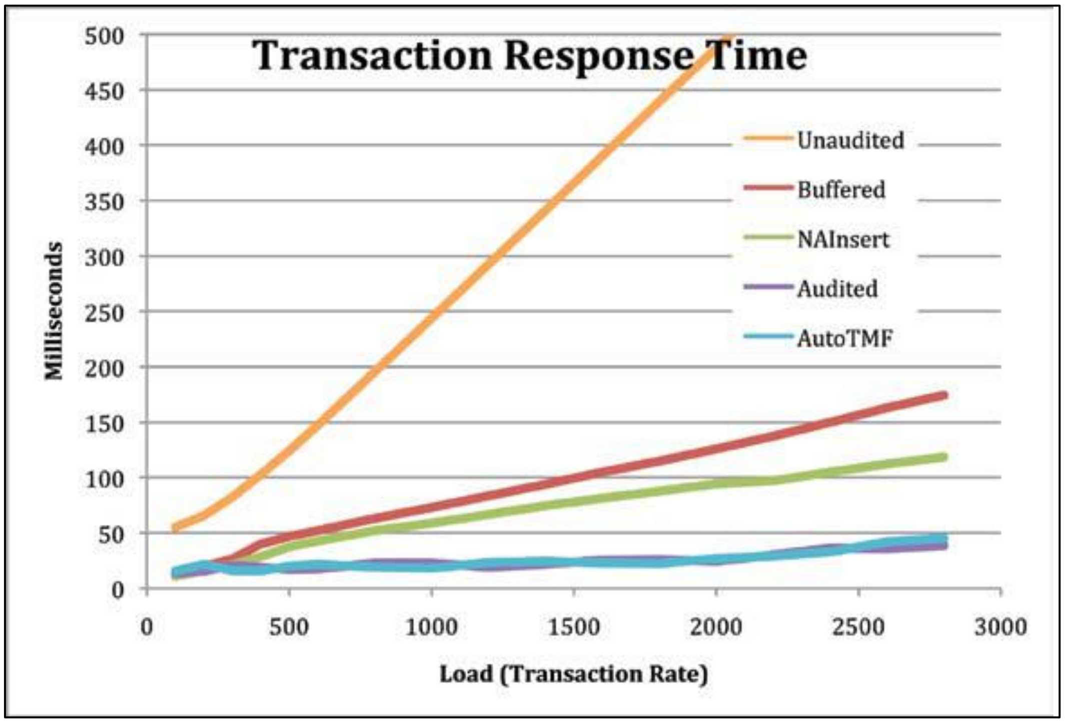


Figure 2 — Transaction Response Time

As confirmed by the performance analysis, the net result of these TMF/DP2 enhancements turns this myth on its head. If you want your application to perform well, and leverage the full capacity of your NonStop system, then you should use TMF audited files and tables.

MYTH: Using TMF Will Require an Application Rewrite

TRUTH: The second objection most often raised about using audited data is that *the use of TMF will require an application rewrite*. For applications that do not currently use audited data, HPE NonStop AutoTMF software automatically provides TMF transaction protection without requiring any change to the application code or logic. AutoTMF works as effectively and efficiently as explicit use of TMF in terms of performance, online backup and recovery, and capturing database changes. AutoTMF is used in hundreds of customer applications. Some adopted AutoTMF to support TMF-based data replication such as RDF or HPE Shadowbase software, and many use AutoTMF to improve application performance.

MYTH: Since AutoTMF is an Extra Step in the Transaction Workflow, it Degrades Performance

TRUTH: Both non-audited data collection and AutoTMF use intercept technology to implement their key functions; however, far more work is performed in the non-audited intercept than in the AutoTMF intercept. If a non-audited data replication solution is being used, then *every* data change (insert, update, or delete) requires an extra I/O to the data replication processes and/or non-audited change data log files. This step adds *considerable* overhead to the non-audited file I/O pathlength and latency for the application, and significantly degrades system and application performance (more than any additional overhead introduced by AutoTMF).

Using AutoTMF as an intercept method does not add more overhead than using other intercept methods to log non-audited I/O data changes. (However, there is no such thing as a free lunch.) The key is to understand what happens after the intercept is invoked. AutoTMF simply makes sure that a TMF transaction is active when one needs to be. It is the file system (and TMF) that process the data collection by using caching; as previously explained, and this sequence is very efficient and fault-tolerant.

Appendix A – Detailed Intercept Technology Data Flows: Non-audited I/O vs. AutoTMF/TMF-protected Files addresses this issue in more detail by illustrating and contrasting audited and non-audited architectures.

***MYTH:* Using AutoTMF and TMF Can Lose Data that Needs to be Replicated**

TRUTH: It is the non-audited replication that can lose data that needs to be replicated. This data loss *cannot* happen with AutoTMF and TMF, and for several reasons. When an I/O intercept mechanism is being used to log data changes for non-audited data, it is very easy to miss individual files, rows, or even entire tables when adding the intercept library to new applications or performing application updates. When this omission happens, the source application silently runs, making data changes at the source that are not collected and replicated to the backup database. Then, the databases diverge. The backup database becomes inconsistent with the primary database, and worse, the user has no idea that it is happening. Such database inconsistency cannot occur when using AutoTMF, because all data changes are automatically audited, and thus are collected and replicated. Forgetting to add the AutoTMF intercept does not allow the source application I/O's to occur against the source's audited file, and the problem is easily seen and corrected very quickly. No data loss occurs at the target.

***MYTH:* Logging and Replicating Non-audited I/O is More Reliable**

TRUTH: The extra steps to save non-audited I/O into the replication engine's log files are fraught with additional failure modes, possibly leading to data loss when failures occur (e.g., certain application failures, I/O cancellations, replication process failures, CPU failures, even system failures and restarts). These data loss scenarios cannot happen when using AutoTMF since TMF is completely fault-tolerant and has been specifically designed, implemented, heavily tested, and is literally used at thousands of sites. If you are still not convinced, please run some system failure tests while using both forms of collection (non-audited intercepting vs AutoTMF and TMF).

***MYTH:* If the Data Replication Engine Fails, it Will Stop my Application**

TRUTH: When any part of a non-audited data replication intercept or solution fails, it may negatively impact your source application's processing, even preventing it from running. This issue is not the case when TMF is used. The replication engine is a completely separate component from the application and TMF subsystem, and failure of any part of the replication engine does not directly impact the application processing.

As a case in point, we are aware of a recent customer situation where a bug in a data replication change data intercept library required an application outage in order to resolve the problem. Furthermore, this customer was running with an active/active business continuity architecture, and the issue required the application on *both* systems to be stopped simultaneously (i.e., a complete application outage). Needless to say, the customer was very unhappy. Such a situation simply cannot arise when TMF auditing is being used, since HPE Shadowbase data replication is totally decoupled from the application. If any updates need to be made to the data replication software, then replication can be stopped in order to effect the change, with no impact to the application. After the update, the replication software will automatically drain the queue of data changes ready to be replicated in order to resynchronize the databases. If an active/active architecture is involved, then the replication software on each node can be independently upgraded at different times (a "rolling" upgrade) so that, again, an application outage is avoided.

***MYTH:* Our Data is Temporal – it is Old News After One or Two Minutes**

Our business continuity plan is to simply fail-over to a standby node, and not to try to recover the failure. Therefore, we see no value in TMF recovery, since we do not want to have to conduct TMF ONLINE DUMPS or AUDITTRAIL DUMPS or perform any other additional work.

TRUTH: The use of TMF auditing and AutoTMF does not mandate the use of TMF recovery, or the use of TMF dumps. An existing business continuity plan that does not rely on these TMF features can be maintained as is.

Additionally, utilizing TMF transactions will keep your individual I/O operations consistent between the base file(s) or base table(s) and the alternate key file(s) or indices. Not using TMF can cause files or tables and their alternate key files or indices to become inconsistent with each other, leading to application failure on takeover (the dreaded file system “Error 59” scenario). Allowing TMF to protect these I/O operations guarantees that the files and tables will not become broken and inconsistent with each other when a failure or failover happens.

MYTH: We Do Not Need TMF, Because We Use BACKUP and RESTORE as our Disaster Recovery Solution

TRUTH: Data that is backed up via BACKUP may be (and usually is) worthless if the source database is OPEN for updating when the BACKUP is taken. This is true even if the BACKUP OPEN option is set to ON. Why? Because the OPEN option only means that the file can be OPEN, and inconsistent, when the BACKUP is taken. Simply put, BACKUP does not resolve file system block splits and joins that can be occurring when the BACKUP occurs. RESTOREing that file or table later will still have the broken blocks and file links in them, rendering the file or table unusable (for example, the file’s “corrupt” bit is set). Worse, this situation will not become apparent until an attempt is made to recover the data from the BACKUP after a disaster (i.e., the backed-up data will be useless and recovery using this data will not be possible).

In order to prevent this catastrophic situation, quiescing the source database (by stopping all UPDATE activity and force-flushing all file blocks) before taking the BACKUP may be an option. However, this is an out-of-date solution which will cause a costly customer outage for the duration of the backup and the time taken to restart operations. It is also unnecessary since TMF has a perfect solution to this problem that allows the database to remain online for updating while the backup is taken so that it can be restored consistently later on. No application outage is needed. It is called TMF ONLINE DUMPS, and it resolves the inconsistencies that can normally occur with regular BACKUP operations!

MYTH: AutoTMF and TMF Require More Disk Space and Change Data Retention Than a Non-audited Architecture Requires

TRUTH: In general, the disk space requirements for TMF and the Audit Trails are similar to the needs of the non-audited solution. Regardless of the collection method used, sufficient (and persistent) disk space must be allocated to contain the change data that needs to be replicated to the target system, and the files that hold the change data need to remain available until that data is replicated successfully. The TMF Audit Trail requirements to satisfy this need should not be substantially different from the disk needs that the non-audited solution requires.

MYTH: AutoTMF License Fees are an Added Expense

TRUTH: Yes, AutoTMF is an add-on product; however, the license cost of AutoTMF and, for example, HPE Shadowbase data replication, is competitive with the license costs of other competing replication products. However, the key point is that AutoTMF should more than offset this *cost* by improved performance and by eliminating the potential risk involved with inconsistent data, data loss, and broken file links, which do not occur in a TMF-protected database, but can occur in a non-audited environment.

MYTH: AutoTMF and TMF Does Not Preserve Database Consistency

TRUTH: Another TMF benefit that must not be overlooked is that of *data integrity* (consistency). Without TMF and its inherent ACID[7] transaction semantics, applications may make data changes that are not fully completed, especially if a data change requires multiple database operations, and/or is split across several applications or sub-routines. Without the use of transactions, recovery from failures in such circumstances must be written into the application and is complex and error-prone, and is very likely to leave the database in an inconsistent state (which can be very costly). By using TMF and audited data, data integrity and error recovery is completely handled by TMF; by definition, a TMF transaction has ACID properties, and in this case,

[7]Atomicity, Consistency, Isolation, Durability (ACID) are the foundational properties of transactional database operations. Paul J. Holenstein, Dr. Bruce Holenstein, Dr. Bill Highleyman, *Breaking the Availability Barrier III: Active/Active Systems in Practice*, Chapter 16 (AuthorHouse, 2007).

is Atomic, meaning either all of the database operations are completed or none of them are, so that the database is always in a consistent state.

More specifically, with the default AutoTMF configuration, database consistency is no worse than the original non-audited application. But AutoTMF allows the customer to modify only the application components that make transactionally-related updates. If a program performs its own BEGINTRANSACTION and ENDTRANSACTION calls, AutoTMF automatically eliminates its own transaction calls. All other programs can use the audited database with no changes.

***MYTH:* Knowledgeable Professional Services Personnel are Unavailable in my Region to Assist with Implementing and Deploying a TMF-based Solution**

TRUTH: HPE and select partners have implemented and deployed hundreds of AutoTMF and TMF-based replication solutions for a wide variety of customers, and have provided global and regional training to enable the customer and reseller staff to maintain these solutions. Please contact your HPE account team or Gravic for more information.

***MYTH:* My Non-audited Application and Database are "Old" and "Legacy?" No Way!**

TRUTH: Yes, a non-audited application and database that cannot guarantee data integrity and consistency is not following current best-practices for relational database management systems (RDBMS). But, such an application may be very costly to rewrite or replace. AutoTMF provides a practical solution to incrementally upgrade the application, adding transactional semantics one program at a time.

Newer and younger management teams tend to view a non-audited application and database that cannot maintain (or guarantee) data integrity and consistency as "antiquated" or "legacy" – out of touch with current best-practices for mission-critical RDBMS. Such a view may lead to replacement of the application, DBMS, or even staff, with something "newer and better." Using TMF eliminates these concerns and offers these features now.

***MYTH:* AutoTMF and TMF Does Not Future-proof Applications**

TRUTH: Future technical improvements in NonStop data protection will require TMF auditing to be in place in order to leverage these advanced capabilities. For example, HPE Shadowbase software has implemented a new and unique Zero Data Loss (ZDL) capability to further protect customer data. This technology ensures that customers' data changes are safe-stored on a target system before the source data changes are allowed to commit. Any subsequent failure of the source system will not lose any critical data, regardless of the type of failure that occurs at the source system, datacenter, or communications network. This additional capability is simply not available without the use of TMF auditing.

Summary

Far from being an impediment, having a TMF audited database not only offers significant operational, reliability, and data integrity advantages, it also improves overall system performance and can lead to improved capacity utilization. The migration from a non-audited to an audited application is very simply made using the facilities provided by AutoTMF, and introduces no significant overhead. The myths and other objections raised against the use of TMF auditing and AutoTMF simply do not stand up to scrutiny.

- The use of TMF and AutoTMF does not impact performance; in fact, in most cases, it dramatically improves it.
- The use of AutoTMF does not require an application rewrite, nor typically any application modifications at all.
- The use of TMF does not require use of TMF recovery or the taking/management of TMF dumps.
- The use of AutoTMF is not expensive in comparison to the potential costs of inconsistent or missing data.
- The use of TMF ensures database consistency and brings an unaudited database up-to-date to the best practices available for DBMS data management.
    - Therefore, it may help to save one’s application (or job) and database from being replaced by ‘something’ more modern.
- The use of TMF helps to future-proof applications, enabling the exploitation of new capabilities (such as Zero Data Loss (ZDL)).

Thus, the path to improving your overall non-audited environment runs through a TMF implementation, and the existence of applications using non-audited data is not an impediment to the deployment of an HPE Shadowbase audit-based data replication solution.

We feel strongly about helping companies improve their NonStop applications and databases with auditing, so along with our HPE colleagues, we will gladly assist with running a trial or Proof-of-Concept with your application and your data, in your IT environment.

Seeing is believing!

More details about the benefits of using TMF, the performance analysis, and the use of AutoTMF can be found in the article, *Best Practices: Using TMF to Implement Business Continuity/Disaster Recovery, written* by Richard Carr of Carr Scott Software Inc., and a TMF survey article, *Boosting Performance with Every Transaction,* by Charles Johnson (former *Distinguished Technologist* and *TMF Elder* at Tandem/Compaq/HPE). We are grateful to Richard for allowing us to reference his article.[8]

[8] For more information, please visit: CarrScott.com

Appendices

Appendix A – Detailed Intercept Technology Data Flows: Non-audited I/O vs. AutoTMF/TMF-protected Files

I/O Event Processing in Non-audited Application Environments (Without Data Replication)

Figure 3 depicts a non-audited I/O event being created and processed.

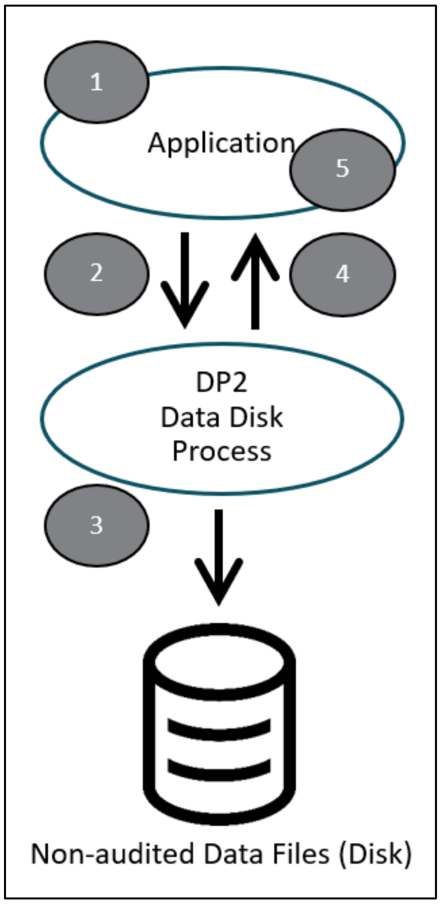


Figure 3 — Non-audited I/O Without a Data Replication Engine

In Figure 3, the non-audited I/O processing steps are:

1. Application: INSERT record INTO file
2. DP2 Data Disk Process receives INSERT
3. DP2 Data Disk Process caches the INSERT and flushes the INSERT to a NA Data File (Disk)
4. DP2 Data Disk Process sends response to Application
5. Application receives response

Non-audited Application Environment Processing I/O Events Using an Intercept Library and Data Replication

Figure 4 extends the previous figure by depicting a non-audited I/O event being captured and subsequent data replication to a backup database. The replication engine is capturing the I/O event, typically safe-storing it into persistent storage such as a disk-based log file, or into an in-memory buffer (the in-memory buffer may represent a single point of failure if it is not protected from loss).

Note that this extensive sequence has many steps, each requiring CPU workload. It is also important to note that these I/O's are often waited, meaning, they are sequential and cannot be parallelized. This exacerbates the slow performance because it introduces queueing time. This architecture is also difficult to scale.

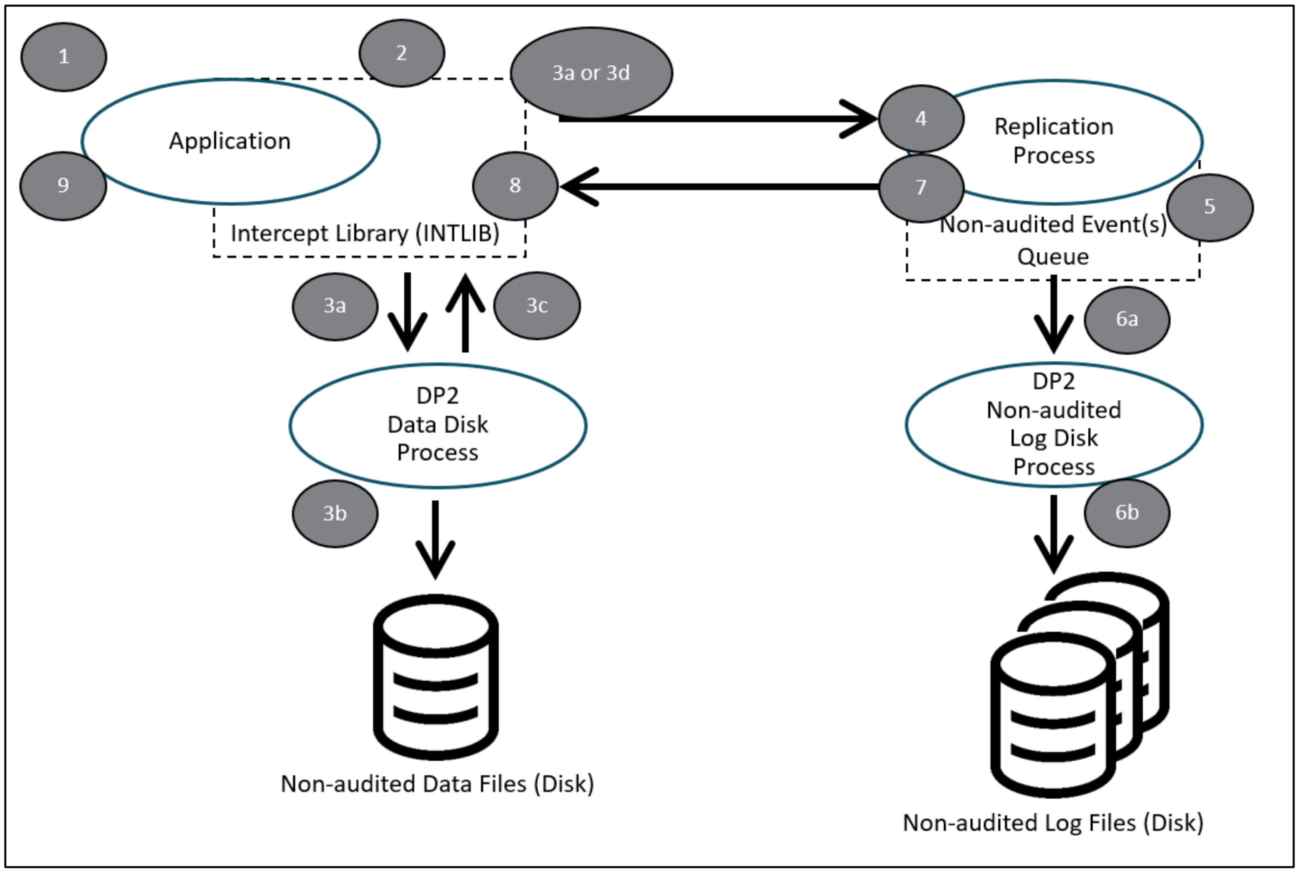

1
2
3a or 3d
Application
9
8
Intercept Library (INTLIB)
4
Replication Process
7
5
Non-audited Event(s) Queue
3a
3c
6a
DP2 Data Disk Process
DP2 Non-audited Log Disk Process
3b
6b
Non-audited Data Files (Disk)
Non-audited Log Files (Disk)

Figure 4 — Non-audited I/O using a separate replication process to flush I/O's to a replication log file

In Figure 4, the non-audited I/O steps are:

1. Application: INSERT record INTO file
2. INTLIB receives INSERT
3. INTLIB
   a. INTLIB sends INSERT to DP2 Data Disk Process (note that this may be done serially, first, or in parallel with the steps 3a-4 (sending the INSERT to the Replication Process).
   b. DP2 Data Disk Process caches the INSERT and flushes the INSERT to a NA Data File (Disk)
   c. DP2 Data Disk Process sends INSERT response back to INTLIB
   d. INTLIB sends INSERT to Replication
4. Replication receives INSERT
5. **Option 1:**
   a. Replication queues INSERT
6. **Option 2:**
   a. Replication flushes INSERT to DP2 NA Log Disk Process
   b. DP2 NA Log Disk Process caches the INSERT and flushes the INSERT to a NA Log File (Disk)
7. Replication sends response back to INTLIB
8. INTLIB receives response & sends response back to the Application

9. Application receives response

Non-audited Application Environment Processing I/O Events, Using Either an Application or an Intercept Library, and Data Replication

Figure 5 contrasts Figure 4, illustrating a more uncommon architecture that captures non-audited I/O events. In this figure, either an intercept library or even the application itself performs the extra steps that capture the I/O event which is subsequently replicated.

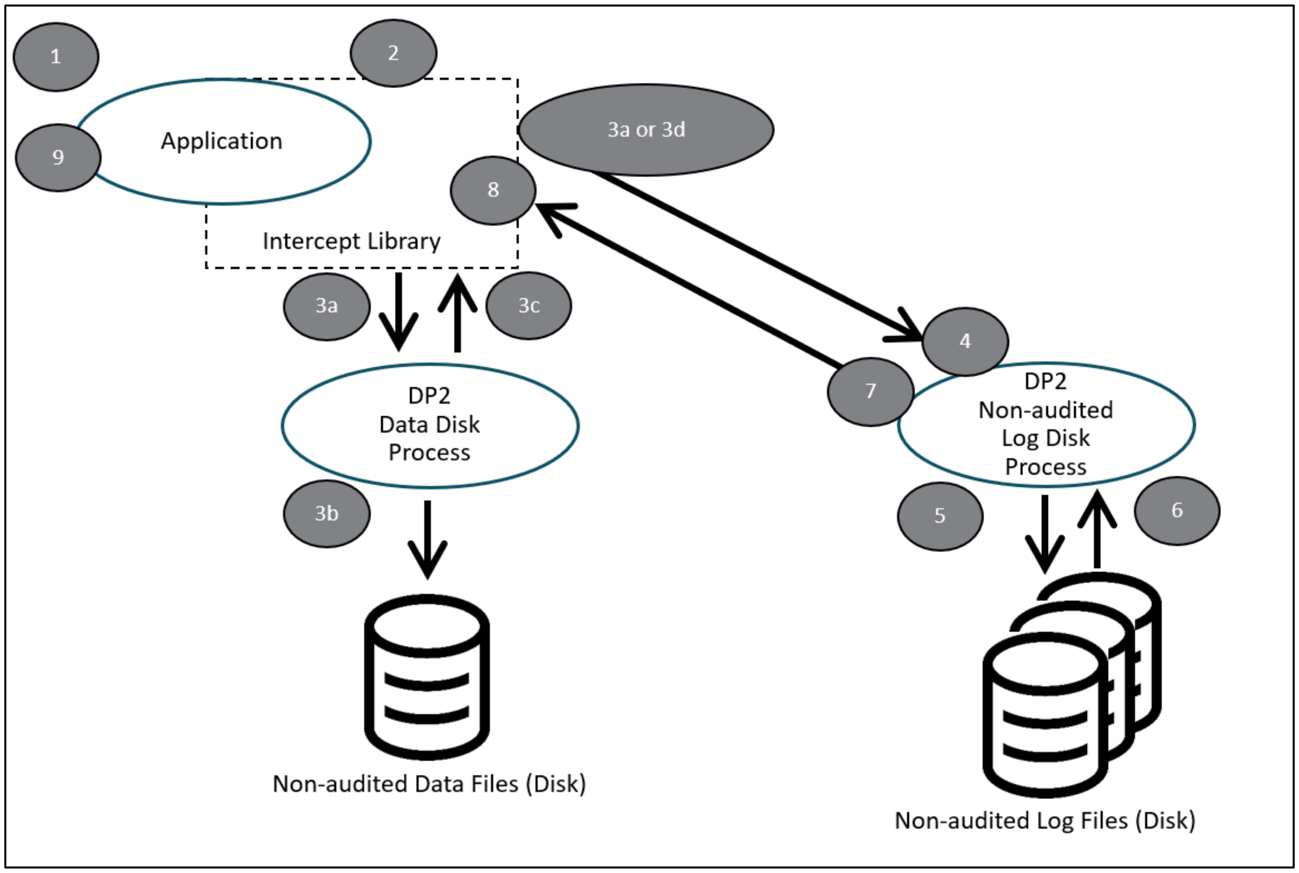

1
2
Application
9
3a or 3d
8
Intercept Library
3a
3c
4
7
DP2 Data Disk Process
DP2 Non-audited Log Disk Process
3b
5
6
Non-audited Data Files (Disk)
Non-audited Log Files (Disk)

Figure 5 — Non-audited I/O writing directly to a replication log file

In Figure 5, the non-audited I/O steps are:

1. Application: INSERT record INTO file
2. INTLIB receives INSERT
3. INTLIB
    a. Send INSERT to DP2 Data Disk Process (note that this may be done serially, first, or in parallel with the steps 3a-4, sending the INSERT to the DP2 Non-audited Log Disk Process).
    b. Flush INSERT to Data Disk
    c. INSERT response back to INTLIB
    d. Either queue or flush INSERT to NA Log Disk Process
4. NA Log Disk Process receives INSERT
5. NA Log Disk Process flushes INSERT to NA Log Disk File
6. NA Log Disk Process receives INSERT response
7. NA Log Disk Process responds to INTLIB
8. INTLIB receives response & sends response to the Application
9. Application receives response

Capturing I/O Events in Audited Application Environments

Figure 6 depicts an audited file I/O event being created, captured by AutoTMF in the form of a transaction, and then safe-stored on disk. AutoTMF manages its audit trail disk processes, which “wrap” around BEGIN… ENDTRANSACTION I/O event boundaries, creating a transaction based on (a) captured I/O event(s). This transaction contains the change data that is subsequently safe-stored in Audit Trail disk files.

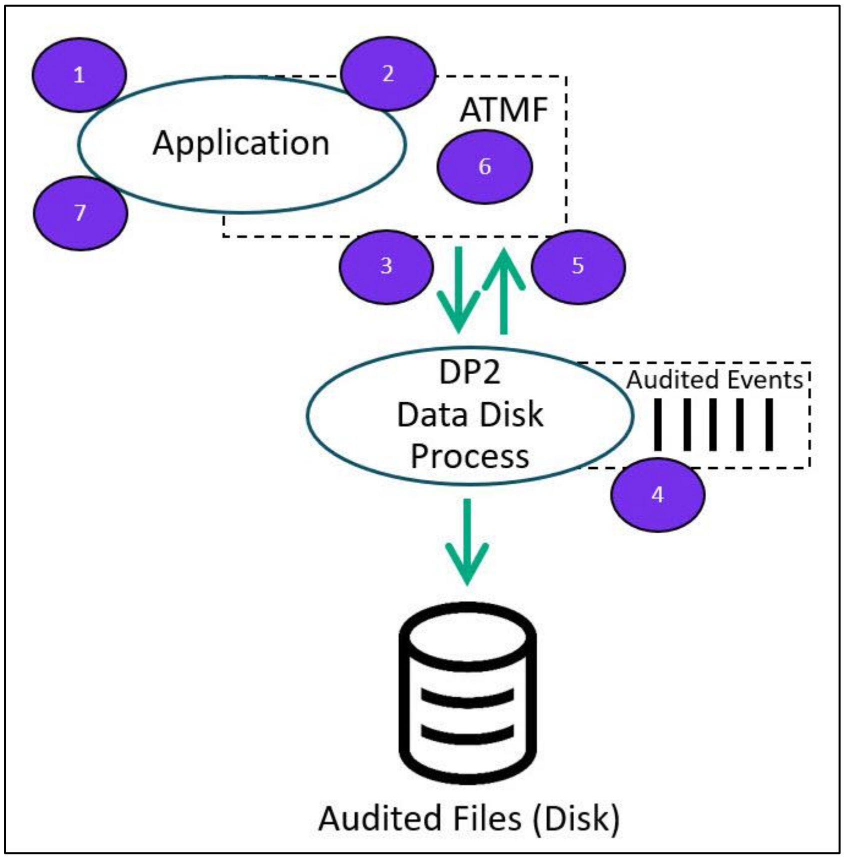


Figure 6 — Audited I/O architecture using AutoTMF (ATMF) to allow TMF protection of the file

In Figure 6, the audited I/O processing steps are:

1. Application: INSERT record INTO file
2. ATMF intercepts INSERT
3. ATMF uses existing TMF transaction (or creates one if it does not already exist) and sends INSERT to DP2 Data Disk Process
4. DP2 Data Disk Process caches the INSERT and queues the INSERT with other Audited Events
5. DP2 Data Disk Process sends response to ATMF
6. ATMF receives response & sends response to the Application
7. Application receives response

I/O Event Pathlength in Audited Application Environments

Figure 7 depicts the total pathlength for audited I/O replication. The application receives and immediately starts capturing and processing data changes instead of waiting for each and every I/O event to be captured and completely safe-stored in the non-audited environment. Application processing is occasionally waited until after TMF performs a COMMIT, however, this latency is often negligible because TMF has been optimized for HPE NonStop COMMIT transactional processing.

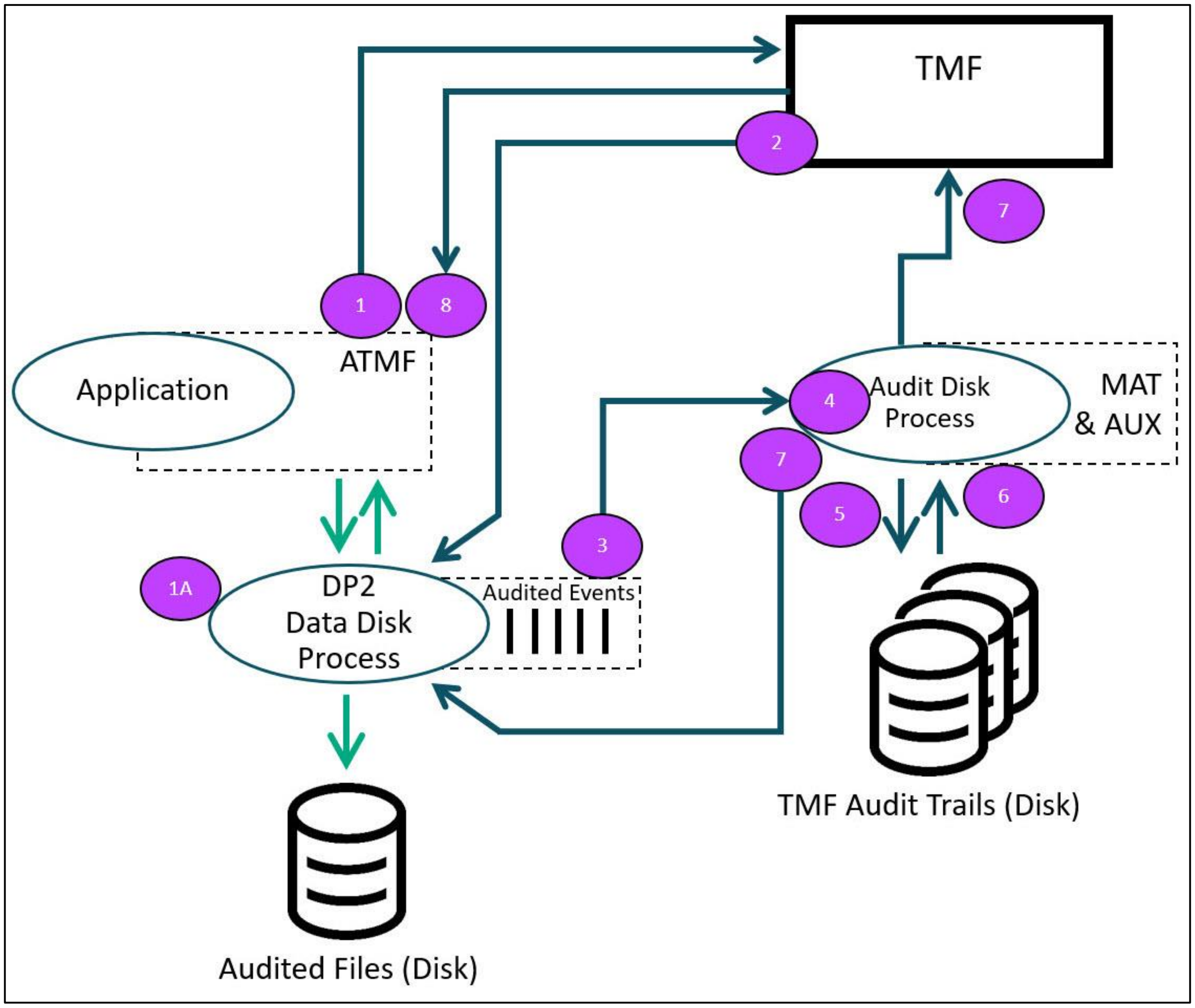


Figure 7 — Pathlength: audited architecture using AutoTMF

In the figure, audited I/O commit steps are:

1. AutoTMF calls COMMIT
2. TMF sends commit signal to DP2 Data Disk Process
3. DP2 Data Disk Process flushes Audited Events to Audit Disk Process
4. Audit Disk Process receives Audited Events
5. Audit Disk Process flushes Audited Events to a TMF Audit Trail Disk
6. TMF Audit Trail Disk sends response to Audit Disk Process
7. Audit Disk Process sends response back to TMF & the DP2 Data Disk Process
8. TMF sends COMMIT COMPLETE message to AutoTMF

Why Isn't Data Replication Shown Here?
We do not show data replication of the captured I/O events in the above figures. The process is similar for either the non-audited replication engine or the audited replication engine. Contact the authors for more information.

International Partner Information

Global

Hewlett Packard Enterprise
6280 America Center Drive
San Jose, CA 95002
USA
Tel: +1.800.607.3567
www.hpe.com

Japan

High Availability Systems Co. Ltd
MS Shibaura Bldg.
4-13-23 Shibaura
Minato-ku, Tokyo 108-0023
Japan
Tel: +81 3 5730 8870
Fax: +81 3 5730 8629
www.ha-sys.co.jp

Gravic, Inc. Contact Information

17 General Warren Blvd.
Malvern, PA 19355-1245
USA
Tel: +1.610.647.6250
Fax: +1.610.647.7958
www.shadowbasesoftware.com
Email Sales: shadowbase@gravic.com
Email Support: sbsupport@gravic.com

Hewlett Packard Enterprise Business Partner Information

Hewlett Packard Enterprise directly sells and supports Shadowbase Solutions under the name ***HPE Shadowbase***. For more information, please contact your local HPE account team or visit our website.

Copyright and Trademark Information