Adabas Transaction Manager

Adabas Transaction Manager Operations Guide

Version 8.2.2

March 2013

Adabas Transaction Manager

This document applies to Adabas Transaction Manager Version 8.2.2.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ATM-PROGRAMMING-822-20130315

Table of Contents

Preface

This document provides information related to Adabas Transaction Manager operations.

The following topics are provided:

1 Startup and Termination

Startup Processing

In a production environment, the various Adabas and Adabas Transaction Manager components should normally be started and allowed to initialize, one after the other, in the following order:

- The Adabas database which contains the Adabas System Coordinator's configuration file (regardless of its `ADARUN DTP=` parameter setting)
- The Adabas System Coordinator daemon within which the TM runs as a service.
- Adabas databases that run with `ADARUN DTP=RM` (referred to as Adabas RMs).
- Adabas databases that run with `ADARUN DTP=NO`.
- Applications

As each Adabas RM starts, recovery is negotiated with the local TM for any transactions that may have been incomplete when the Adabas RM last terminated (normally or abnormally).

Whether the TM starts before an Adabas RM or after, each incomplete transaction is finally committed or backed out at the earliest opportunity; that is, when all the necessary information is available to the TM at the root of the transaction and all the Adabas RMs, peer TMs and any other vendor transaction frameworks that were involved in the transaction are available.

When an Adabas RM starts before the TM, and it has prepared transactions that have not completed, full use of that database is possible only when the TM has initialized and carried out restart processing for the incomplete transactions. Until this occurs, the resources of the incomplete transactions are held, and the owner of the transaction is not permitted to work on that database.

If the TM encounters a serious error while trying to resolve an incomplete transaction during restart, details are reported to the operator and to `DDPRINT` for up to 100 problematic transactions. Thereafter, errors are reported in the `DDPRINT` dataset only.

Termination Processing

Adabas RM closedown procedure

The following procedure should be followed when closing down an Adabas RM:

1. Make sure that any TM to which the Adabas RM is connected to is active (connection messages are written to the Adabas RM output file).
2. Issue an `ADAEND` command to the Adabas RM.

   If the Adabas RM is not currently taking part in distributed transactions, Adabas will action the `ADAEND` request without further intervention from Adabas Transaction Manager.

   If the Adabas RM is currently taking part in distributed transactions, the following process will occur:

   - The Adabas RM will prompt the appropriate TM to quiesce all outstanding transactions which involve this Adabas RM. An ATM171 message acknowledging this request is written to the Adabas RM output file.
   - The TM quiesces all appropriate transactions and signals the Adabas RM that the `ADAEND` process can continue. An ATM173 message indicating a successful quiesce is written to the Adabas RM output file.
   - The Adabas RM closes down.

TM closedown procedure

The following procedure should be followed when closing down a TM:

1. Issue an `ATM END` command to the Adabas System Coordinator daemon in which the TM is running.

   If there are no incomplete distributed transactions, the TM will close down immediately.

   If there are incomplete distributed transactions, the following process will occur:

   - An ATM103 message will be issued for each of the first five incomplete transactions.
   - This ATM103 message will be repeated at 60 second intervals until all transactions are complete.
   - When all transactions are complete, the TM will close down.

Caution: Once a TM service closes down within a System Coordinator daemon it can only be restarted by stopping and starting the System Coordinator daemon.

2 Operator Commands

ATM operator commands can be issued by the operator in the normal way or from the Online Services application.

The following operator commands are supported by an ATM transaction manager:

Command	Description
`ATM DSTAT`	In response to this command, the TM will display transaction statistics.
`ATM END`	This command requests an orderly shutdown of the TM. New distributed transactions will be prevented from starting. In-flight distributed transactions will be allowed to complete, at which point the TM will terminate.
`ATM HALT`	This command requests an immediate shutdown of the TM. New distributed transactions will be prevented from starting. In-flight distributed transactions will be terminated without waiting for completion. Recovery of incomplete transactions occurs during restart.
`ATM RSTAT`	In response to this command, the TM resets all of its statistical counts to zero. A message appears on the console confirming that statistics have been reset.
`ATM STOPU='jobname'`	This command asks the TM to perform a “stop” operation for all transactions belonging to clients with the given jobname. The TM will attempt to terminate such transactions by backing them out, or (if the commit decision has already been taken) committing them, then it will discard its knowledge of any transactions that were successfully terminated. An ATM082 message will appear on the console to indicate that this operation was requested by the operator.

3 Restart and Recovery

Recovery Records

The TM records details of incomplete, prepared transactions in its recovery file. You can use the Online Services application to check for incomplete transactions in the system.

Caution: The recovery file is a critical resource where the TM persists information about itself and the systems and applications it is working amongst. You must *not*

- change the Node ID of the Adabas System Coordinator daemon within which the TM runs as a service;
- change any of the following transaction manager parameter values: `TMETDATA`, `TMSYNCMGR`, or `TMTCIDPREF`.

If you wish to change these things you must make sure there are no incomplete transactions, all ET data are preserved so it is possible to recreate it again, etc.

Suspect Transaction Records

The TM uses suspect transaction records (STJ) in the recovery file to record all known details of incomplete transactions that have been purged from the system as a result of intervention by the operator or database administrator.

Incomplete transactions can be purged as follows:

- using the Stop Transaction function provided by Online Services, or
- as a result of a forced restart using the runtime parameter `TMRESTART`.

Online Services can be used to browse through the suspect transaction records..

Alternatively, you may use the sample program ATMSPRNT in the supplied JOBS library to produce a readable printout of the suspect transaction records. See Print Suspect Transaction Records for more information. Use the comments in the job when modifying it to conform to site requirements.

There is no automatic housekeeping of suspect transaction records. It is intended for emergency use only. The database administrator should purge these records from time to time, after making sure that the information contained in it is no longer required.

Adabas Resource Locks

Distributed transactions require more management to make sure they are conducted safely so they will have a tendency to take longer to complete than transactions that are not distributed. Distributed transactions take longer because Adabas must preserve information about held records in its Work when the transaction is prepared (for restart/recovery purposes). This can have knock-on effects on hold queue sizes etc, which need to be considered.

Handling unplanned outages

Transaction Manager outage and failover

Quite obviously, if the transaction management service fails (this usually implies the System Co-ordinator daemon has also failed) it should be restarted immediately. In most systems automatic restart management makes sure this happens immediately and the TM negotiates recovery with all databases, etc automatically and immediately without intervention. Therefore the general rule is to restart TM as soon as possible after a failure.

Unplanned outage in a single system

When TM becomes unavailable applications will run according to their *Continuous operation mode* setting. This may mean their transactions continue to run or that errors are issued. As soon as the TM returns restart/recovery takes place as appropriate and things quickly return to normal.

Unplanned outage in a multi-system

Multiple TMs running as peers in a group across multi-systems can collaborate and provide failover assistance to each other. In-flight transactions being managed by a TM that suffers outage are automatically and dynamically dispersed among TM peers based upon the failover capabilities of the applications that are running

Some applications support dynamic transaction routing in a multi-systems environment (such as CICS/PLEX). These applications can be configured to dynamically adopt failover by dispersing client sessions across peers in the same or other systems. The TM peers react automatically to client sessions being dispersed by negotiating transfer of transaction management to the TM that is local to where the client session is itself transferred (this demands a TM runs in each eligible system).

Other applications either do not support or are not configured for dynamic transaction routing so they will run according to their *Continuous operation mode* setting as previously described, and peer TM may takeover some clean-up if that is possible. Similar to single-systems, the general rule

is that a failed TM service should be restarted immediately. When it is restarted it will negotiate its restart/recovery with its TM peers and continue normally.

Note: Given that the general rule is to restart a failed TM immediately (and that most large systems do this automatically anyway) then a default action is that no TM peers will pre-empt any failover duties until 60 seconds have passed. This gives time for a failed TM to resume its duties without others doing unnecessary failover work.

Undetected Database Restarts

With or without Adabas Transaction Manager it can happen that:

- A client session has an in-flight transaction with Adabas.
- Adabas is recycled without the client session being made aware (for example it may be busy communicating with a different database).
- During this recycle Adabas undoes (backs out) the in-flight transaction.
- When Adabas returns, the client performs more modifications (not knowing the previous were undone).
- The client commits (applies) the latest modifications.

The result is inconsistency from the perspective of the client but not from the perspective of Adabas.

This situation can arise where a) ETID are not used and b) Adabas runs without `OPENRQ=YES`. In this situation the client session will not receive a response code 9 so the recycle of Adabas will go undetected.

To avoid this situation (with or without using Adabas Transaction Manager):

- Set the ADARUN parameter `OPENRQ=YES` for all databases, so that a restarted database will give response code 9, subcode 66, indicating that a new OP is required.
- Use ETIDs for all client sessions, so that “backout during system open” sets a pending response 9 for the affected session.

Recovery with the CICS Resource Manager Interface

For a system which has been configured to use the CICS Resource Manager Interface, the following recovery process occurs at CICS startup (or soon after):

1. The Adabas Transaction Manager CICS re-synchronization driver program (ATMRMIRS) obtains from the local TM a list of all prepared (but incomplete) transactions that were controlled by this CICS system.
2. CICS is then instructed to re-synchronize each of these transactions.

3. During this process, CICS indicates whether each of these transactions should be backed out or unconditionally committed.
4. When the last incomplete transaction has been processed, the TM writes a console message indicating that the re-synchronization process is complete.

In order to re-synchronize incomplete transactions in this way, CICS logging must be active and CICS must be warm started. If CICS logging is not in use or if CICS is cold started when there are incomplete transactions in the system, transaction integrity cannot be guaranteed. Consult your CICS documentation for more information.

Recovery with RRMS

When RRMS is already active at TM start (this is the normal case), the TM re-synchronizes in cooperation with RRMS to resolve any incomplete transactions that were under RRMS control.

When RRMS is unavailable at TM start, the TM issues a warning message to the console and waits until RRMS becomes available. Then it re-synchronizes.

If a critical component of RRMS becomes unavailable while the TM is operating, a warning message is issued to the console. In some cases, the TM is able to continue processing and initiates re-synchronization processing as soon as the missing component is reactivated.

4 Transaction Processing

Transaction Coordination Priority

Transactions can be controlled externally by a client-side transaction coordinator (e.g. CICS RMI), or a host system transaction coordinator (e.g. RRMS). When both types of coordinators are in operation, the client-side coordinator will take transactional precedence..

For example, when a transaction is under the control of CICS RMI, coordination of that transaction will be performed by CICS RMI even if the TM is running RRMS-enabled.

Prepared Transactions

A prepared transaction is one where, during the completion stage of a distributed transaction, all participating resource managers have acknowledged successfully processing the prepare phase (phase-1) of a two-phase commit process. For an Adabas RM this means all necessary transaction resources have been retained.

Note: A prepared transaction is virtually “frozen” until completed - it is not subject to the normal Adabas timeout rules associated with transactions and nonactivity.

5 Adabas Databases

The ADARUN DTP parameter indicates whether or not a database is capable of full participation in Distributed Transaction Processing. Normally, when a database is started with DTP=RM, it is immediately "signed on" to the Transaction Manager for Distributed Transaction Processing. This means that the Transaction Manager uses two-phase commit protocol to guarantee the integrity of distributed transactions that modify this database.

There might be occasions, however, when the process of "signing on" for DTP cannot be completed immediately, perhaps because of a planned or unplanned outage of another component that is itself going through startup processing at the time. During this transient period, Adabas Transaction Manager ensures uninterrupted operation by treating databases that have not signed on for DTP as if they were running wth DTP=NO. In these circumstances, a commit operation is applied to all "unsigned on" databases in turn immediately after DTP commit has been completed for all databases in the transaction that are signed on, by means of serial ET commands. At some later point this transient "not signed on" period ends because the sign-on eventually succeeds, Adabas Transaction Manager recognizes the change, and from that point the database is treated as a DTP=RM database.

In a multi-system environment, it is possible to run completely separate System Coordinator groups in the separate systems. For example, a "production" group might run on system A, while a "test" group might run on system B. The DTP=RM databases used by the "production" environment would be executing outside the scope of the "test" System Coordinator group. If an application in the "test" environment modifies a DTP=RM database in the "production" environment, Adabas Transaction Manager recognizes that the database is executing outside the scope of the current System Coordinator group, and it manages the database (for the "test" client) as if it were running with DTP=NO.

6 Client Sessions

Client Session Memory Requirements

Note: For information about the client-related memory requirements of the Adabas System Coordinator in the application address space, refer to the *Adabas System Coordinator* documentation.

The additional memory requirement per client session for the Adabas Transaction Manager client proxy is approximately:

- 1024 bytes for session maintenance
- plus 268 bytes for session statistics maintenance
- plus 16 times the value of the `MaximumNumberOfDatabases` runtime control

Bear in mind that certain settings of the Natural `ADAMODE` parameter cause Natural to execute two sessions in parallel for each terminal user. This increases the effective number of clients in the client address space.

7 Using Adabas VSAM Bridge

If a CICS system is to support both the Adabas VSAM Bridge and the ATM Resource Manager Interface implementation, the Adabas Task-Related User Exit (TRUE) must be enabled before the VSAM Bridge is activated.