

Adabas Transaction Manager

Adabas Transaction Manager Operations Guide

Version 7.5.1

September 2009

Adabas Transaction Manager

This document applies to Adabas Transaction Manager Version 7.5.1 and to all subsequent releases.

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

Copyright © Software AG 2009. All rights reserved.

The name Software AG, webMethods and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

1 Adabas Transaction Manager Operations Guide	1
2 ATM Database Management	3
ATM Database Usage and Restrictions	4
ET Data Storage	4
Pool/Queue Usage Control	5
Diagnostic Logging	5
Excluding DBMSs from Global Transaction Processing	7
Disengaging a Database from Two-Phase Commit Processing	7
3 Required Datasets	9
Transaction Manager Daemon JCL Requirements	10
Database JCL Requirements	10
Cluster Database JCL Requirements	11
4 ATM Startup and Termination	13
Startup Processing	14
Termination Processing	14
5 Operator Commands	17
6 Restart and Recovery	19
ATM Recovery Records	20
Suspect Transaction Journal	20
Adabas Resource Locks	21
ATM Transaction Completion	21
Recovery with the CICS Syncpoint Manager	22
Recovery with RRMS	22
7 Syncpoint Processing Options	23
Overview of Syncpoint Processing Options	24
Overriding the Current TRANMODE Option	25
Adabas Commands and External Syncpoints	27
8 Transaction Processing	29
Transaction Coordination Priority	30
Heuristic Completion of Prepared Transactions	30
9 User Sessions	33
User Session Identification using Communication IDs	34
User Session Memory Requirements	34
10 Using Adabas VSAM Bridge	35
Index	37

1 Adabas Transaction Manager Operations Guide

This document provides information related to Adabas Transaction Manager operations.

The following topics are provided:

- **ATM Database Management**
- **Required Datasets**
- **ATM Startup and Termination**
- **Operator Commands**
- **Restart and Recovery**
- **Syncpoint Processing**
- **Transaction Processing**
- **User Sessions**
- **Using Adabas VSAM Bridge**

2 ATM Database Management

- ATM Database Usage and Restrictions 4
- ET Data Storage 4
- Pool/Queue Usage Control 5
- Diagnostic Logging 5
- Excluding DBMSs from Global Transaction Processing 7
- Disengaging a Database from Two-Phase Commit Processing 7

ATM Database Usage and Restrictions

The ATM daemon runs as a special kind of database nucleus for which the following restrictions apply:

- no files may be loaded to this database except as detailed in the installation process.
- no application program or user session may issue Adabas calls to this database.

At version 7.5, the recovery file, suspect transaction journal and ET data file must reside in the ATM daemon's own database.

If it becomes necessary to perform maintenance of the ATM daemon's database (for example, to reorder the checkpoint file), you must first shut down the system. Then carry out the maintenance normally in either single-user or multi-user mode. You must first, temporarily, set the ADARUN parameter `DTP=NO`.



Note: Any utility jobs run against the ATM daemon database must use an Adabas link module that does not use a TM proxy or the ATM job parameter `ATM=OFF` must be set.

Once you have defined an ATM daemon database, its Database ID should remain fixed. If you need to change it, you must first ensure that there are no incomplete transactions anywhere in your system. Follow the normal procedure for closing down the ATM daemon and, if appropriate, its recovery database. Then close down all application/client environments in which a TM proxy is active. Now you can change the Database ID. Next, follow the normal procedure for activating ATM, then restart the application/client environments.

ET Data Storage

- [ET Data Storage in the ATM Daemon's Database](#)
- [ET Data Storage with External Transaction Coordinators](#)

ET Data Storage in the ATM Daemon's Database

When Adabas Transaction Manager is in use with the runtime parameter setting `ADARUN TMETDATA=ATM`, ET data is stored in and read from the ATM daemon's database.

If your applications need their current ET data to be established in the ATM daemon's database before they can execute, refer to section Copy ET Data for more information.

ET Data Storage with External Transaction Coordinators

When running with the CICS Syncpoint Manager or Recoverable Resource Management Services (RRMS), it is not possible to synchronize the storage of ET data when an unsolicited syncpoint occurs, because CICS and RRMS syncpoints have no knowledge of ET data.

If an application stores ET data and runs in a CICS/RMI or RRMS environment, you can ensure that the storing of its ET data is synchronized with the two-phase commit process by conforming to the following rules:

- Any syncpoint for which ET data is to be stored must be triggered by an ET or CL command.
- The ET or CL command that triggers the syncpoint must also supply the ET data; that is, if the application issues a series of ET commands to different databases, the first ET must supply the ET data.

In an IMS/TM system whose transactions are coordinated by RRMS, it is not possible to store ET data synchronously with an RRMS syncpoint. IMS allows an RRMS commit syncpoint to take place only at the successful completion of message processing, and this syncpoint cannot be triggered by an ET or CL command.

Pool/Queue Usage Control

Most pools and queues used by ATM will expand dynamically as required. In other cases, to help avoid filling a pool or queue, ATM issues a warning message to the operator and to DDPRINT when the internal pool or queue area becomes 85% full (rounded down). The high-water marks can be displayed using the Online Services application.

Diagnostic Logging

- [Using ATMLOG Datasets](#)

- [Using TMPLOG Dataset](#)

Using ATMLOG Datasets

You can optionally write Adabas Transaction Manager diagnostic log information to sequential log datasets. For MVS systems (z/OS and OS/390), the characteristics of the log datasets are:

- sequential organization
- fixed-block records
- record length of 256 bytes
- block size 5120 bytes

For BS2000 systems the ATMLOG datasets are SAM files with variable-length record format.

Use of the log datasets is determined by the ADARUN parameter `TMLLOG` and the operator command `TM LOG`.

The log datasets `ATMLOG1` and `ATMLOG2` must be defined in the ATM daemon startup job control. It is recommended that you specify `DCB=BUFNO=1` for these datasets on MVS systems (OS/390 and z/OS), to avoid the possibility of Sx37 abends during termination of the ATM daemon or switching of log datasets.

A sample Natural job `ATMLPRNT` is provided in the supplied `JOBS` library for use in producing a readable report from a log dataset. Use the comments in the job when modifying it to conform to site requirements. The content of the report is undocumented, and subject to change. A log report might be requested by Software AGs support for problem diagnosis.

Using TMPLOG Dataset

In MVS systems (OS/390 or z/OS), any batch jobs that use a TM proxy cause the following message to appear in its JES message log:

```
IEC130I TMPLOG DD STATEMENT MISSING
```

This message can be ignored. Alternatively, you can suppress it by including the following JCL statement in the job step:

```
//TMPLOG DD DUMMY
```

The `TMPLOG DD` name can be used to provide a dataset for a diagnostic log that details activity within the TM proxy.



Caution: Use this diagnostic log only after consulting with Software AG support.

Excluding DBMSs from Global Transaction Processing

Rigorous management of global transactions inevitably creates overhead, which can be minimized by careful exclusion of certain databases. For example:

- A development database generally does not require the same guarantees of transaction integrity as a production database and can therefore be excluded from two-phase commit processing.
- System files generally do not require the same guarantees of transaction integrity as application data files. If you maintain application data files and system files in different databases, those containing system files can often be excluded from two-phase commit processing.

The `ADARUN DTP` parameter is used to include an Adabas database (`DTP=RM`) or exclude it from (`DTP=NO`) participation in two-phase commit processing. Remember, however, that ATM generates `ET` or `BT` commands to changed databases that run with `DTP=NO`, when a user's global transaction terminates. For more information about the way ATM handles commands directed at databases running with `DTP=NO`, see the section Adabas Transactional Commands.

Disengaging a Database from Two-Phase Commit Processing

If a database running with the `ADARUN` parameter setting `DTP=RM` is terminated, a message is issued by the ATM daemon indicating

- that an RM has signed off;
- whether or not the RM had unresolved transactions; and
- if so, whether the unresolved transactions were prepared or unprepared.

If you restart the database with `DTP=NO`, you must also specify `IGNDTP=YES` for its first execution. However, you may lose the integrity of incomplete prepared transactions if you restart the database with `DTP=NO` and `IGNDTP=YES`. To avoid this, you must resolve any incomplete transactions before switching to `DTP=NO`.

If you restart a database with a different `DTP` parameter value, the change will not be recognized by any TM proxy components, for users who are already in session with that database. This could cause errors if such a user tries to change the database. However, new user sessions will recognize the new parameter value, and users who close the database and issue a new `OP` command will be able to carry on processing normally.

If you need to change a database's `DTP` parameter, the safest procedure is as follows:

- ensure that the database is closed cleanly, with no incomplete global transactions in flight;
- restart all application/client environments which use the database;

- restart the database.

3 Required Datasets

- Transaction Manager Daemon JCL Requirements 10
- Database JCL Requirements 10
- Cluster Database JCL Requirements 11

Transaction Manager Daemon JCL Requirements

The following table shows the dataset definitions that are essential for correct execution of the ATM daemon. Other datasets, such as command log, may be specified. For details of these, refer to the Adabas documentation.

Dataset	Link Name	More Information
Load library		ADA743 library with LX06 update; COR742 library, update 5 or above; ATM75x library
Associator	DDASSORn	
Data Storage	DDDATARn	
Work	DDWORKR1	
ATM log	ATMLOG1/2	Optional, depending on TMLOG parameter value
ADARUN parameters	SYSDTA/ DDCARD	DTP=TM, and other operational parameters
ADARUN messages	SYSOUT/ DDPRINT	

Database JCL Requirements

The following table shows the dataset definitions that are essential for correct execution of a single database that is to execute as a resource manager, under the transactional control of ATM. Other datasets, such as command log, may be specified. For details of these, refer to the Adabas documentation.

Dataset	Link Name	More Information
Load library		ADA743 library; COR742 library, update 5 or above; ATM75x library
Associator	DDASSORn	
Data Storage	DDDATARn	
Work	DDWORKR1	
ADARUN parameters	SYSDTA/ DDCARD	DTP=RM, and other operational parameters
ADARUN messages	SYSOUT/ DDPRINT	

Cluster Database JCL Requirements

The following table shows the dataset definitions that are essential for correct execution of an Adabas Cluster Services or Adabas Parallel Services database that is to execute as a resource manager, under the transactional control of ATM. Other datasets, such as command log, may be specified. For details of these, refer to the Adabas documentation.

Dataset	Link Name	More Information
Load library		ADA743 library with LX06 update; COR742 library, update 5 or above; ATM75x library
Associator	DDASSORn	
Data Storage	DDDATARn	
Work	DDWORKR1	
Work 4	DDWORKR4	Recoverable information about global transactions
ADARUN parameters	SYSDTA/ DDCARD	DTP=RM, and other operational parameters
ADARUN messages	SYSOUT/ DDPRINT	

4 ATM Startup and Termination

- Startup Processing 14
- Termination Processing 14

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 System Coordinator daemon that runs under the same Adabas SVC or ID table as the ATM daemon
- The ATM daemon
- Databases that run with `DTP=RM`

If the local Adabas System Coordinator daemon is not active when the ATM daemon starts, the ATM daemon will issue a warning message periodically, until the SYSCO daemon is found to be active. During this time, ATM will not be available for use by applications.

As each RM database starts, its local ATM daemon is notified, so that it can coordinate restart of any incomplete global transactions in which the RM database is involved.

Whether the ATM daemon initializes first or not, each incomplete transaction is finally committed or backed out at the earliest opportunity; that is, when all the necessary information is available to the ATM daemon at the root of the transaction and all the databases, ATM daemons, and any external coordinator involved in the transaction are available.

If for some reason a database running with `ADARUN DTP=RM` initializes before the ATM daemon, and the database has prepared transactions that have not completed, full use of that database is possible only when the ATM daemon 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 (that is, the owner's ETID, if one was assigned) is not permitted to work on that database.

If the ATM daemon 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

- [Closedown Procedure for a DTP=RM Database](#)

- [Closedown Procedure for an ATM Daemon](#)

Closedown Procedure for a DTP=RM Database

The following procedure should be followed when closing down a database for which DTP=RM is in effect:


1. Make sure that any ATM daemon to which the RM is signed on is active.
2. Enter an ADAEND command for the RM database. If the database is currently taking part in global transactions, it will ask the appropriate ATM daemon(s) to quiesce these transactions.
3. If the database issues an ATM073 message, an error occurred in communication with the ATM daemon indicated by the message, and the ADAEND process will not complete correctly. Check that the ATM daemon is active and retry the ADAEND. If it still fails, go to step 7.
4. An ATM daemon will issue console messages indicating whether or not it was able to quiesce global transactions for this database. An ATM068 message is always issued. ATM069 and ATM071 messages indicate an error which is likely to prevent the ADAEND process from completing, and which probably requires manual intervention. ATM070, ATM072 and ATM078 messages all indicate pending completion for some global transactions, probably due to a temporary condition. Check the availability and status of external transaction coordinators, remote ATM daemons and Net-work connections. The ATM daemon will retry the operation at intervals of about a minute. If none of these messages follows the ATM068 message, the operation was successful. Once this status has been reached by all ATM daemons which were asked to quiesce transactions, the database will terminate as soon as all users reach ET status. An ATM126 console message will be issued by the ATM daemon(s) at this point.
5. If the database does not terminate quickly, you might choose to issue an operator command to reduce its TT value. Before doing this, you should read and understand the following steps, and also the description of [Heuristic Completion of Transactions](#).
6. If the database still does not terminate quickly, monitor for the messages described in step 4 - these will re-appear approximately every minute, as long as the ATM daemon is unable to quiesce global transactions for the database.
7. If operator intervention is possible, use Online Services to check for incomplete transactions on the database. (use function Transaction Manager Daemon Information > Display Known Databases, then use the line command L). If necessary you can force completion of a transaction using the Stop function.

It might be necessary to use the Transfer to STJ option in the case of a transaction that is controlled by some other ATM daemon or an external coordinator.




Caution: Use of the STJ option for this purpose may lead to an inconsistent result.

8. If you need to close down the database immediately and leave it in a clean state, issue a HALT command.

 **Caution:** Use of the `HALT` command can cause heuristic completion of prepared transactions which can lead to inconsistent results.


9. If you need to close down the database immediately, and it is acceptable to preserve incomplete transactions for completion during restart, do not use the `HALT` command; instead, use `CANCEL` - either the Adabas operator command or the operating system command.

 **Caution:** Do not delay this decision unnecessarily, otherwise the nucleus might heuristically complete prepared transactions whose `TT` limits have expired which could lead to inconsistent results.

Closedown Procedure for an ATM Daemon

The following procedure should be followed when closing down an ATM daemon:

1. If you do not intend to restart the daemon immediately, you should first make sure that there are no global transactions active that have reached 'prepared' status on any local `DTP=RM` database. You can use Online Services (`SYSATM`) to display the global transaction queue to determine if there are active global transactions which have reached 'prepared' status. If so, it may be impossible to `ADAEND` a `DTP=RM` database while the daemon is not executing, without incurring heuristic termination of some transactions which could cause inconsistent results.
2. Issue the `TM END` command to the ATM daemon.
3. If the daemon is unable to terminate within about a minute, because of incomplete global transactions, it will issue an `ATM103` message to the console, indicating that `TM END` is pending. You might then choose to issue `TM HALT`. The daemon should then terminate immediately, leaving any unresolved work for completion during restart.

 **Note:** This pending incomplete work will not prevent utilities from running against the ATM daemon's database - the `WORK` dataset(s) will be clean.

On normal termination, the ATM daemon writes runtime statistics to the `DDPRINT` dataset.

4. Close down the local Adabas System Coordinator daemon, if desired.

5 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 Adabas Transaction Manager daemon:

Command	Description
TM DSTAT	In response to this command, the ATM daemon will display, on the console log and in DDPRINT, the current high-water marks and other statistics. Refer to the descriptions of messages ATM113 , ATM114 , ATM116 and ATM117 for details.
TM END	This command requests an orderly shutdown of the ATM daemon. ATM prevents any new global transactions from starting. It waits until any in-flight global transactions have completed, then terminates.
TM FEOFLOG	In response to this command, the ATM daemon closes the log dataset that is currently in use and opens the other log dataset. A message appears on the console indicating the dataset currently in use. If logging to the ATMLOG datasets is not active, the command will have no effect.
TM HALT	This command requests an immediate shutdown of the ATM daemon. ATM prevents any new global transactions from starting. It terminates without waiting for any in-flight global transactions to complete. Recovery of incomplete transactions occurs during restart.
TM LOG	In response to this command, the ATM daemon starts writing log records to the ATMLOG dataset. A message appears on the console indicating that logging is active. This command has no effect if the daemon was started with ADARUN TMLOG=NEVER.
TM NOLOG	In response to this command, the ATM daemon stops writing log records to the ATMLOG dataset. A message appears on the console indicating that logging is no longer active.
TM RSTAT	In response to this command, the ATM daemon resets all of its statistical counts to zero. A message appears on the console confirming that statistics have been reset.
TM STOPU='jobname'	This command asks the ATM daemon to perform a stop users operation for all users with the given jobname. The ATM daemon will attempt to terminate any incomplete

Command	Description
	transactions belonging to these users by backing them out, or (if the commit decision has already been taken) committing them, then it will discard its knowledge of any such users who now have no incomplete transaction. An ATM082 message will appear on the console to indicate that this operation was requested by the operator.

6 Restart and Recovery

▪ ATM Recovery Records	20
▪ Suspect Transaction Journal	20
▪ Adabas Resource Locks	21
▪ ATM Transaction Completion	21
▪ Recovery with the CICS Syncpoint Manager	22
▪ Recovery with RRMS	22

ATM Recovery Records

The ATM daemon records details of incomplete, prepared transactions in the recovery record file. Whenever a transaction is completed, its recovery record is deleted. If the contents of this file are lost at a time when incomplete, prepared transactions exist in the system, ATM is not able to guarantee the integrity of those transactions.

You can use the Online Services application to check for incomplete transactions in the system.



Caution: When incomplete transactions exist, you must *not*

- change the Database ID of an ATM daemon;
- change any of the following ATM ADARUN parameter values: TMDYNTCIDS, TMETDATA, TMSYNCMGR, or TMTCIDPREF.

Suspect Transaction Journal

ATM uses the suspect transaction journal (STJ) 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 User 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 contents of the STJ file.

Alternatively, you may use the sample program `ATMSPRNT` in the supplied `JOBS` library to produce a readable printout of the contents of the STJ file. See `Print STJ File` for more information. Use the comments in the job when modifying it to conform to site requirements.

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

Adabas Resource Locks

During the life of a transaction, an application gains ownership of certain database resources associated with the transaction. For example, records that are changed, or new and old unique descriptor values. Adabas locks these resources against use by other users or applications until the transaction is completed; that is, committed or backed out.

In the case of a global transaction, Adabas secures this information, together with the necessary recovery information, in its Work dataset. The information is not discarded until the end of the two-phase commit process for the owning transaction. It also survives database restart if the transaction has successfully completed the prepare phase.

ATM Transaction Completion

When a database nucleus with the runtime parameter `DTP=RM` is started, it signs on to ATM and provides the details of any incomplete global transactions.

ATM then attempts to complete each of the transactions by instructing each relevant database to commit its transaction or roll back its changes, as appropriate. If any of the incomplete transactions have branches in other systems, the partner ATM daemons in those systems are also instructed to commit or roll back, as appropriate.

Meanwhile, the resources that were changed by the incomplete transaction remain unavailable to other users.

When started, an ATM daemon obtains, from its recovery file and from any partner ATM daemons in other systems, details of any global transactions that were incomplete when it last terminated. It then attempts to complete each of these transactions by instructing each relevant database to commit its transaction or roll back its changes, and each partner ATM to commit or roll back its transaction branches, as appropriate.

The integrity of global transaction is thus secured across restarts of critical components.

ATM does not decide whether to commit or back out a prepared transaction that is controlled by an external transaction coordinator. Any such transaction remains in doubt until the external coordinator resolves it.

Recovery with the CICS Syncpoint Manager

For a CICS system in which an Adabas TRUE interface is used, the TRUE is enabled during CICS startup. If the system has been configured to use the RMI, the program that enables the TRUE interface to attempt to coordinate recovery with the local ATM daemon. If the ATM daemon is inactive, the recovery process is deferred until ATM starts up.

The recovery process is the same, whether it occurs during CICS startup or ATM startup:

1. The ATM CICS Resync Driver program obtains from the local ATM daemon a list of all incomplete (but already prepared) 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 tells ATM whether each transaction should be backed out or unconditionally committed.
4. When the last incomplete transaction has been processed, the ATM daemon writes a message to the console indicating that the re-synchronize process is complete.

In order to re-synchronize incomplete transactions between CICS and ATM, 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.

Recovery with RRMS

If RRMS is already active when the ATM daemon starts up, which is normally the case, ATM re-synchronizes in cooperation with RRMS to resolve any incomplete transactions that were under RRMS control.

If RRMS is unavailable when the ATM daemon starts, the ATM daemon 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 ATM is operating, a warning message is issued to the console. In some cases, ATM is able to continue processing and initiates re-synchronization processing as soon as the missing component is reactivated.

7 Syncpoint Processing Options

- Overview of Syncpoint Processing Options 24
- Overriding the Current TRANMODE Option 25
- Adabas Commands and External Syncpoints 27

This section describes the available Syncpoint processing options.

Overview of Syncpoint Processing Options

The normal rules of transactional programming under CICS and other TP systems are different from those that are familiar to Adabas programmers. Specifically, in a standard CICS application, a screen I/O normally means that pending changes are committed, and locks are freed, whereas Adabas allows a transaction to span screen I/Os. Indeed, Natural utilities such as SYSMAIN could behave differently when the ATM CICS Resource Manager Interface is in use, because they are written in such a way that they expect transactions to remain open across screen I/Os.

With the advent of ATM, this difference in programming styles led to two requirements for the ATM CICS RMI implementation:

- it should be possible for pending Adabas changes to remain uncommitted across pseudo-conversational task end;
- it should be possible to have ISNs released from held status, automatically, when a CICS syncpoint occurs.

The first requirement makes the introduction of ATM easier for sites that have taken advantage of the possibility of keeping Adabas transactions open across screen I/Os. It also allows Natural utilities to execute as before. The second brings Adabas behavior into line with standard behavior in CICS and other TP environments, and is ATM's normal mode of operation. The two are, of course, incompatible, and are therefore implemented as options through the `TRANMODE` job parameter.

TRANMODE=MESSAGE Option

This is the default option, and defines ATM's standard way of processing syncpoints. Processing of a message always terminates with a syncpoint. Normally this means that a screen I/O causes pending changes to be committed. If an external transaction coordinator is in control of a transaction, an `ET` or `CL` command will trigger a commit syncpoint by the external coordinator; a `BT` command, or an `OP` to a changed database, will trigger a rollback syncpoint by the external coordinator. Held ISNs will be released, or will remain held, according to the setting of the extended hold option, and any `P` or `M` command options. An unsolicited syncpoint from an external transaction coordinator will cause ATM to commit or back out pending changes to all databases and release all held ISNs. In any case, the commit process will be synchronized for all `DTP=RM` databases; changes to other databases will be committed or backed out after completion of the syncpoint processing.

TRANMODE=DYNAMIC Option

This option allows existing Adabas applications (including Natural utilities such as SYSMAIN) to execute under the ATM CICS RMI without being affected by the unsolicited CICS syncpoints that occur at pseudo-conversational task end. ATM will honor all rollback syncpoints, whether

they originate from `BT` or `OP` commands, from a `CICS` command, or from `CICS` itself. `ATM` will also honor commit syncpoints triggered by `ET` or `CL` commands, but it will ignore other commit syncpoints.

This option is appropriate for the Natural utilities, and for applications which keep transactions open across screen I/O operations. However, it might not be suitable for applications that execute under the `CICS` RMI and change both Adabas and non-Adabas resources. If such an application encounters an unsolicited commit syncpoint (when a screen I/O occurs, for example), its non-Adabas changes will be committed, but the Adabas changes will remain uncommitted until an `ET` or `CL` command is executed. That is, all syncpoints will continue to affect other resource managers (such as `DB2`) exactly as they did before, regardless of the behavior of `ATM`.



Note: If you use `TRANMODE=MESSAGE` under the `CICS` RMI, and execute Natural using an `ADAMODE` setting that causes Natural to execute two parallel Adabas sessions, you must make sure that Natural zaps `NA44231` and `NA45010` are applied, otherwise repeated response code 9, subcode 97 will occur, and other problems may follow. These zaps are both included in Natural version 3.1.6. You must also ensure that Natural's system session always begins with an `OP` command, otherwise the same problems will occur; you can do this by always supplying a non-blank `ETID` or by using the `DBOPEN` parameter.

The `TRANMODE` parameter has no effect for `IMS/TM` systems whose transactions are coordinated by `RRMS`, and whose local `ATM` runs with `TMSYNCMGR=RRMS`.

Overriding the Current `TRANMODE` Option

The `TRANMODE` option specified in the job parameters takes effect, by default, for all users executing in matching jobs. However, a user can dynamically change the `TRANMODE` setting for the user's own session by calling a utility program. A Natural `CALLNAT` program and a command-level `CICS` assembler program are provided for this purpose.

The supplied programs, `PRMORIDE` (Natural) and `ATMORID` (assembler) are functionally similar. They can be executed during the execution of a `CICS` task. They override the default `TRANMODE` setting for the user of the current `CICS` task. The new setting will remain in effect until it is overridden by another execution of either utility program, or until the user finally stops work or logs onto `CICS` again. The Natural programs can be used in a similar way in environments other than `CICS`.



Note: In `TP` systems where Adabas users are identified by Terminal ID, and where a Terminal ID can be re-used by different users in succession, a new user can inherit the `TRANMODE` setting that was in effect for the previous user of the Terminal ID. Therefore, if the override facility is used in such a system, it should always be executed at the start of a new session, to ensure that the correct `TRANMODE` is set, rather than assuming that the default setting will be in effect.

The assembler program, *ATMORID*, is executed by CICS LINK. It expects a COMMAREA containing the following fields:

```

RETCODE DS F Output parameter - return code
ADANAME DS CL8 Input parameter - name of the Adabas link
*           module through which the caller's commands
*           are processed
PARMSTR DS CL16 Input parameter - new parameter string
DIAGS1  DS F Output parameter - error diagnostics
DIAGS2  DS F Output parameter - error diagnostics
DIAGS3  DS F Output parameter - error diagnostics
    
```

The parameter string must be one of the following:

```

TRANMODE=DYNAMIC
TRANMODE=MESSAGE
TRANMODE=?
    
```

The *TRANMODE* parameter replaces the *SYNCMODE* parameter of ATM version 1.2. For the sake of compatibility with programs that were written to operate with ATM version 1.2, *SYNCMODE* parameter overrides are also accepted as follows:

- *SYNCMODE=ALL* is equivalent to *TRANMODE=MESSAGE*
- *SYNCMODE=ADABAS* is equivalent to *TRANMODE=DYNAMIC*
- *SYNCMODE=?* will return either *SYNCMODE=ALL* or *SYNCMODE=ADABAS*, as appropriate.

After successful execution, the return code will be zero, and the *PARMSTR* field will contain the current *TRANMODE* setting. A return code of 4 indicates a CICS LINK error. Other values are ATM error codes, as described in the section Messages and Codes.

The subprogram *PRMORIDE* is the Natural equivalent of *ATMORID*. It expects two parameters:

- The parameter string (*TRANMODE=MESSAGE*, *TRANMODE=DYNAMIC*, or *TRANMODE=?*) (A16)
- A return code field (N4)

The following is a sample program that calls *PRMORIDE*:

```

DEFINE DATA
LOCAL
1 #PARM-ORIDE (A16)
1 #PARM-RET-CODE (N4)
END-DEFINE
INPUT #PARM-ORIDE(AD=I'_' )
CALLNAT 'PRMORIDE' #PARM-ORIDE #PARM-RET-CODE
WRITE '=' #PARM-RET-CODE
END
    
```

If PRMORIDE detects an error it will set the return code (values and meanings are the same as for ATMORID) and display some diagnostic information on the screen.

Adabas Commands and External Syncpoints

ATM allows Adabas changes to be committed synchronously with non-Adabas changes, by interacting with external transaction coordinators. When an external transaction coordinator takes a syncpoint, ATM ensures that changed Adabas databases take part in the commit or rollback operation. By default, ATM also causes the external transaction coordinator to take a syncpoint whenever it detects that pending changes are to be committed or backed out.

Usually, this means that every ET or CL command causes an external commit syncpoint, and any BT command causes an external rollback syncpoint to take place. However, there are cases in which this behavior might be different from what is required. For example, consider a CICS environment in which a Cobol program changes DB2, then starts a Natural session, expecting Natural to return control before a decision to commit or back out is taken. Natural can issue an ET command during LOGON processing, and CL commands at session end. By default, each of these commands (if issued under the same Communications ID as the Cobol program's commands) would cause a CICS SYNCPOINT to take place, and the first of these would cause the pending DB2 changes to be committed.

Job parameters are provided which can be used to change this behavior. See the descriptions of the SYNCONBT, SYNCONCL, and SYNCONET parameters. In the example described above, it would be appropriate to specify SYNCONCL=NO and SYNCONET=ET, so that the CL and ET commands generated by Natural would not cause a CICS SYNCPOINT.

By default, these job parameters take effect for all users executing in matching jobs. However, a user can dynamically change their values for the user's own session, by calling a utility program. This can be achieved using a supplied Natural CALLNAT program or (in a CICS system) command-level CICS program. The technique is the same as for changing the current TRANMODE setting, and the same Natural and CICS programs are used for the purpose. See [Overriding the Current TRANMODE Option](#) for details. The following parameter strings are accepted by the override programs:

```
SYNCONBT=YES
SYNCONBT=NO
SYNCONBT=?
SYNCONCL=YES
SYNCONCL=NO
SYNCONCL=?
SYNCONET=YES
SYNCONET=NO
SYNCONET=?
```



Note: In TP systems where Adabas users are identified by Terminal ID, and where a Terminal ID can be re-used by different users in succession, a new user can inherit the parameter settings that were in effect for the previous user of the Terminal ID. Therefore, if the override facility is used in such a system, it is a good policy to ensure that it is always executed at the start of a new session, to ensure that the correct setting takes effect, rather than assuming that the default setting will be in effect.

8 Transaction Processing

- Transaction Coordination Priority 30
- Heuristic Completion of Prepared Transactions 30

Transaction Coordination Priority

The ATM instances involved in a transaction can be running with `TMSYNCMGR=RRMS` or `TMSYNCMGR=NONE`.

When the transaction is under the control of a client-side transaction coordinator such as the CICS Syncpoint Manager TMP parameter (`LCLSYNC=YES`), each ATM daemon is aware of this and does not involve RRMS, even if its own ADARUN parameters indicate that its RRMS interface is to be activated (`TMSYNCMGR=RRMS`).

The transaction executing under the CICS/RMI can change Adabas and DB2 databases, for example, and these changes are coordinated by the CICS Syncpoint Manager. For such a transaction, the ATM daemon behaves as if it were running with `TMSYNCMGR=NONE`.

In order for the transactions of a job or IMS/TM system to be coordinated by RRMS, you must set the job parameter `HSTSYNC=YES`, and ATM must run with ADARUN parameter `TMSYNCMGR=RRMS`.

Heuristic Completion of Prepared Transactions

- [When is Heuristic Completion Required?](#)
- [Adabas Termination of a Global Transaction](#)
- [Transaction Timeout \(TT\) Setting](#)
- [Online Save](#)
- [ADAEND Command](#)
- [Internal Synchronized Checkpoint](#)

When is Heuristic Completion Required?

In certain situations, it may be necessary for a resource manager to make a heuristic decision about completing a transaction that it has already prepared.

For example, suppose a resource manager is asked by its local transaction manager to prepare its part of a global transaction initiated by a remote user, and then the network fails. The resource manager would normally lock any resources held by the transaction and wait to receive an instruction from the transaction manager to commit or roll back.

The transaction resources could remain locked for a long time. While the locks remain, the resource manager is unable to perform certain functions, such as a database save.



Note: A prepared transaction is not subject to the normal timeout rules.

Adabas Termination of a Global Transaction

An Adabas nucleus will terminate its part of a global transaction which it has successfully prepared, if all of the following are true:

- an internal ET syncpoint occurs; and the TT time limit currently in effect for the user has expired.

An internal ET syncpoint occurs when:

- an ADAEND command is issued;
- an online save or delta save syncpoint occurs; or
- a SYNCC command is issued.

An Adabas nucleus will also terminate its part of a global transaction if a HALT command is received, or if Work part 4 overflows.

In most of the above cases, the Adabas nucleus decides to commit the local transaction. However, if HALT is issued or Work part 4 overflows, the nucleus backs out the local transaction. In all cases, Adabas writes information about the terminated transaction to its DDPRINT dataset for audit purposes.



Caution: It is strongly recommended not to use the HALT command for an Adabas nucleus running with the parameter setting DTP=RM.

Transaction Timeout (TT) Setting

The risk of heuristic transaction terminations would increase if an operator command were allowed to reduce the TT value (in preparation for an ADAEND command or an online save, for example). For this reason, an Adabas nucleus running with DTP=RM normally ignores a command to reduce its TT value if it has any prepared transactions that have not yet completed. However, if the nucleus has a pending ET syncpoint (at the end of an online save, for example), the new TT value will take effect.

Online Save

If a prepared transaction remains at the end of an online save operation, and has not been completed within a period of 60 seconds plus the value of the TTSYN parameter, measured from the beginning of the transaction, the Adabas nucleus will heuristically back out the transaction, to allow the pending ET syncpoint to complete.

ADAEND Command

When an Adabas nucleus running with `DTP=RM` receives an `ADAEND` command, it prevents new transactions from being prepared or started and asks its local ATM daemon to quiesce any global transactions in which it is involved. If this process is successful, any prepared transactions are completed (committed or backed out) and the `ADAEND` command is then processed. If prepared transactions remain incomplete, they are heuristically terminated when their `TT` time limits expire, and then the `ADAEND` command is processed.

Internal Synchronized Checkpoint

An internal synchronized checkpoint takes place at the end of an online save operation. This requires all users of the nucleus to be at `ET` status simultaneously. No user can begin a new transaction until the checkpoint has completed. The `TTSYN` parameter of `ADASAV` can be used to make a temporary reduction in the nucleus's `TT` value; this causes incomplete transactions to be backed out after the specified time. This logic applies only to unprepared transactions. If `ATM` is in use, there is a possibility for prepared transactions to remain open after the period specified by `TTSYN`. If this is the case, an additional period of approximately 60 seconds is allowed for prepared transactions to complete. Once this extra period has elapsed, the nucleus will heuristically back out any transactions remaining in prepared status. An `ADAN89` message will be written to the console and `DDPRINT` for each such heuristic termination.



Caution: Heuristic termination can result in the loss of transaction integrity. When part of a global transaction is terminated, the global transaction as a whole may have mixed completion, which means that parts of it are committed while other parts are backed out. Mixed completion indicates a loss of global transaction integrity. However, since unilateral local transaction completion applies only to prepared transactions, the likelihood of mixed completion is small.

9 User Sessions

- User Session Identification using Communication IDs 34
- User Session Memory Requirements 34

User Session Identification using Communication IDs

Adabas Transaction Manager identifies a user session by its 28-byte Communications ID.

If a user issues Adabas commands under different Communications IDs, ATM regards these commands as having been issued by different users. For example, this could occur during dynamic transaction routing in a CICS environment if the Adabas System Coordinator is not being used to manage the user sessions.

If the same Communications ID is to be used consecutively in more than one client environment (for example, CICS and batch), the first session must be terminated cleanly before the second is started.

User Session Memory Requirements



Note: For information about the user-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 user for the Adabas Transaction Manager proxy is approximately:

- 1200 bytes
- plus 8 times the value of the `MAXDB` job parameter
- plus 96 times the value of the `LGRECNO` job parameter

The memory management functions of the Adabas System Coordinator might perform some upward rounding when it allocates memory for use by the Adabas Transaction Manager proxy, so the actual memory usage per user could be greater than indicated by the above estimate.

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

A syncpoint operation that occurs under the CICS RMI when an `ET`, `BT`, `OP` or `CL` command is issued, is handled under a shadow User ID, associated with the original user. This, too, effectively increases the number of active users in the CICS address space.

10

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.

Index
