Adabas Transaction Manager Benefits and Features

This section provides an overview of the benefits and features provided by Adabas Transaction Manager.

The Role of the Adabas Transaction Manager

The Adabas Transaction Manager is a server for coordinating distributed transaction processing in distributed Adabas environments. It manages global transactions that are distributed across multiple Adabas databases by coordinating changes to the databases in a seamless, integrated way, using a two-phase commit protocol when necessary.

Adabas Transaction Manager addresses two basic needs:

- the need to deliver industrial strength enterprise objects for widespread commercial use in mainstream, critical business systems, and
- the need to spread the large volumes of data that Adabas customers manage more evenly across the computer(s) and organization.

Adabas customers have significant investments in existing application systems. The Adabas Transaction Manager’s client proxy and Transaction Manager make it possible for these systems to participate in distributed transaction processing transparently.

Global Transactions

- Global Transaction Time Limits
- Coordinating Global Transactions Across Systems

What is a Global Transaction?

A global transaction is a unit of work that involves changes to resources under the control of one or more database management systems (DBMSs) operating in one or more system images. The databases and operating system images can be local or remote. Transaction processing operations can be handled serially or in parallel.

Once Adabas Transaction Manager software has been installed and configured, existing applications can execute global transactions with guaranteed all-or-none integrity. In terms of application logic, nothing changes except that the application may receive new response codes if errors occur that are associated with the management of global transactions. Such error conditions can be handled normally by an error-handling routine. Details of the possible error conditions can be found in the section Messages and Codes.

Global Transaction Status

A user is said to be at global transaction status when there are no uncommitted changes for that user in databases running with runtime parameter setting `DTP=RM`; that is, when the user has no pending changes in databases that are capable of two-phase commit processing.

Global Transaction Time Limits

In the same way that Adabas provides a time limit for local database transactions, Adabas Transaction Manager supports a time limit for global transactions. If a global transaction is incomplete when the time limit expires, Adabas Transaction Manager attempts to complete it by committing it or rolling it back, depending on its status. If the transaction is rolled back, the application receives response code 9 with a suitable subcode notifying it of the backout.

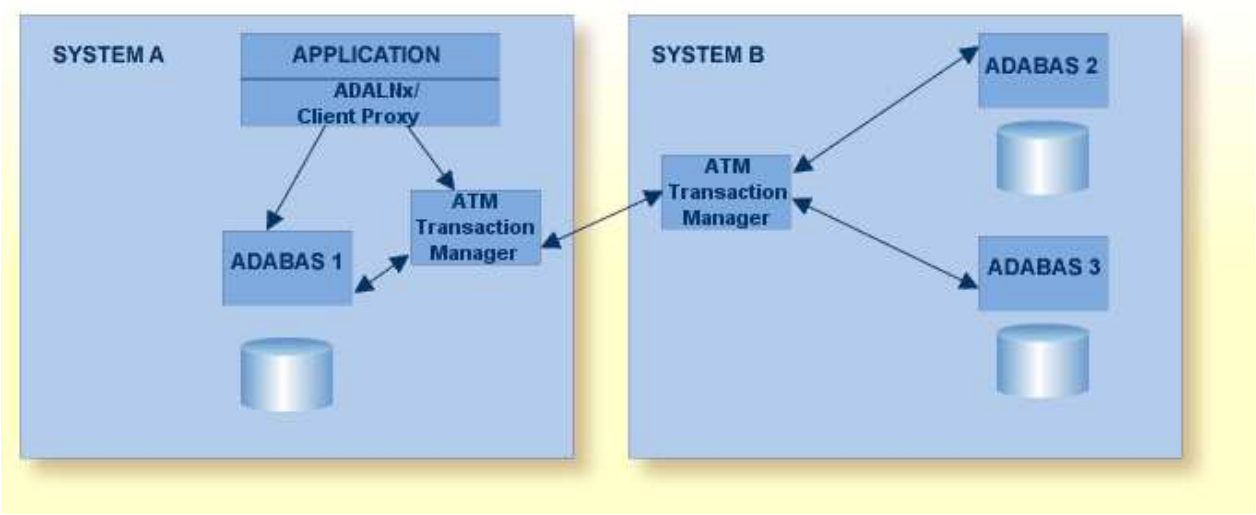
Coordinating Global Transactions Across Systems

If Entire Net-Work is installed, global transactions can cross system boundaries. The ATM transaction manager that represents one operating system image coordinates with one or more other ATM transaction managers, each of which represents another operating system image.

The root ATM transaction manager operating in the system image that is local to the user (application) takes a superior role in coordinating the global transaction; the partner transaction managers operating in other system images that process parts or branches of the global transaction take a subordinate role.

An ATM transaction manager can take a superior role for one transaction while simultaneously taking a subordinate role for another transaction.

Through the ATM client proxy component, the root transaction manager accepts a request from the application to commit. It instructs all Adabas databases in its system image to prepare and it instructs all partner transaction managers to prepare their branches of the global transaction. If all prepare instructions are completed successfully, the root transaction manager instructs each local database and partner transaction manager to commit.



Distributed Transaction Processing

- What is Distributed Transaction Processing?
- Adabas Support for Distributed Transaction Processing

What is Distributed Transaction Processing?

Processing global transactions across multiple local or remote databases and/or system images in parallel is called distributed transaction processing (DTP).

Distributed transaction coordination processing ensures the integrity of global transactions by making it possible for each participating database to process its part of the transaction independently, in parallel with other databases processing their parts of the transaction. Global transactions are secured or rejected as a whole across separately managed resources in two phases:

- In phase one (prepare phase), all databases participating in the global transaction are asked whether the local part of the transaction can be committed.

  During phase one, participating databases must retain all transaction resources to prepare for any event during phase two.

- In phase two (commit or back out phase), when all participating databases have replied, the global transaction is committed if all replies are positive or backed out if any reply is negative.

Each distributed environment includes:

- the application (AP) for which a database transaction is performed;
- the database management systems (DBMSs) called resource managers (RMs) that participate in processing the global transaction;
- a component called a transaction manager (TM) that uses two-phase commit processing to coordinate the activities of RMs in a single operating system image; and

- a component called a communications resource manager (CRM) that coordinates the activities of TMs participating in a global transaction that spans more than one operating system image.

Adabas functions in this scenario as a resource manager.

The transaction manager and communications resource manager roles in this scenario belong to the ATM transaction manager, which processes a global transaction across multiple Adabas databases and, if necessary, across multiple system images using Entire Net-Work to communicate between those images.

Adabas Transaction Manager works in partnership with transaction managers that take a higher-level, controlling role in coordinating global transactions, such as the CICS Syncpoint Manager or the Recoverable Resource Management Services (RRMS) from IBM. In this way, Adabas Transaction Manager participates in global transactions that involve changes to both Adabas and non-Adabas database or file systems such as DB2 or VSAM, within a single operating system image.

As an Entire Net-Work node, each ATM transaction manager is aware of the other ATM transaction managers in the distributed system and the resource managers they coordinate. The ATM transaction managers act in partnership to coordinate distributed transactions. At any time, each ATM transaction manager can account for the status of the global transactions it is coordinating.

Distributed transaction support can be implemented for some applications and not for others. Applications that use an Adabas link module with a transaction manager client proxy can benefit from Adabas Transaction Manager's two-phase commit processing while those without a client proxy continue to execute as before without Adabas Transaction Manager. Additional control is possible through the use of client control parameters.

Adabas Support for Distributed Transaction Processing

Adabas incorporates nucleus functions to support the execution of distributed transaction processing using a two-phase commit protocol that is transparent to existing Adabas application systems and to Natural. The following items are relevant to distributed transaction processing:

- The ADARUN runtime parameters `DTP, LDTP`, and `IGNDTP`.
- The Adabas Work component, part 4 or WORK4 dataset.
- For CICS users, a CICS-controlled interface is provided that conforms to the CICS Resource Manager Interface (RMI) so that Adabas databases can participate in global transactions coordinated by the CICS Syncpoint Manager. The interface issues the appropriate Adabas commands to participate in the two-phase commit protocol.

  See also the section Interfacing with Other DBMSs.
- Similarly, for applications that run in a z/OS system as single-task batch jobs, or under Com-plete or IMS TM, an interface to RRMS is provided. This allows Adabas databases to participate in global transactions coordinated by RRMS.

  See also the section Interfacing with Other DBMSs.

Distributed transaction support is available for `ET` logic users only.

Adabas Transaction Manager Version 8.1 provides distributed transaction support for Adabas Cluster Services and Adabas Parallel Services databases.

Processing Modes

Global transactions may be processed using the following processing modes:

- *distributed transaction mode* using Adabas Transaction Manager and the two-phase commit protocol, or
- *serial mode* using a series of ET and BT commands.
- Distributed Transaction Mode
- Serial Mode

Distributed Transaction Mode

In distributed transaction mode, a two-phase commit protocol is used as described below.

- Phase 1: Prepare
- Phase 2: Commit or Remove
- Example

Phase 1: Prepare

During phase 1, each database changed by a global transaction is asked to prepare its changes. This means that the database and its associated logs must reach a state where the changes can either be committed or removed as requested by the caller. The database must respond to the prepare request, indicating success or failure.

Phase 2: Commit or Remove

When Adabas Transaction Manager has received the results of the prepare requests from all databases participating in the global transaction, it can begin phase 2.

- If any prepare request failed, Adabas Transaction Manager issues a roll back request to each of the databases to remove the changes.
- If all prepare requests were successful, Adabas Transaction Manager issues a commit request to each of the databases to complete the global transaction.

Example

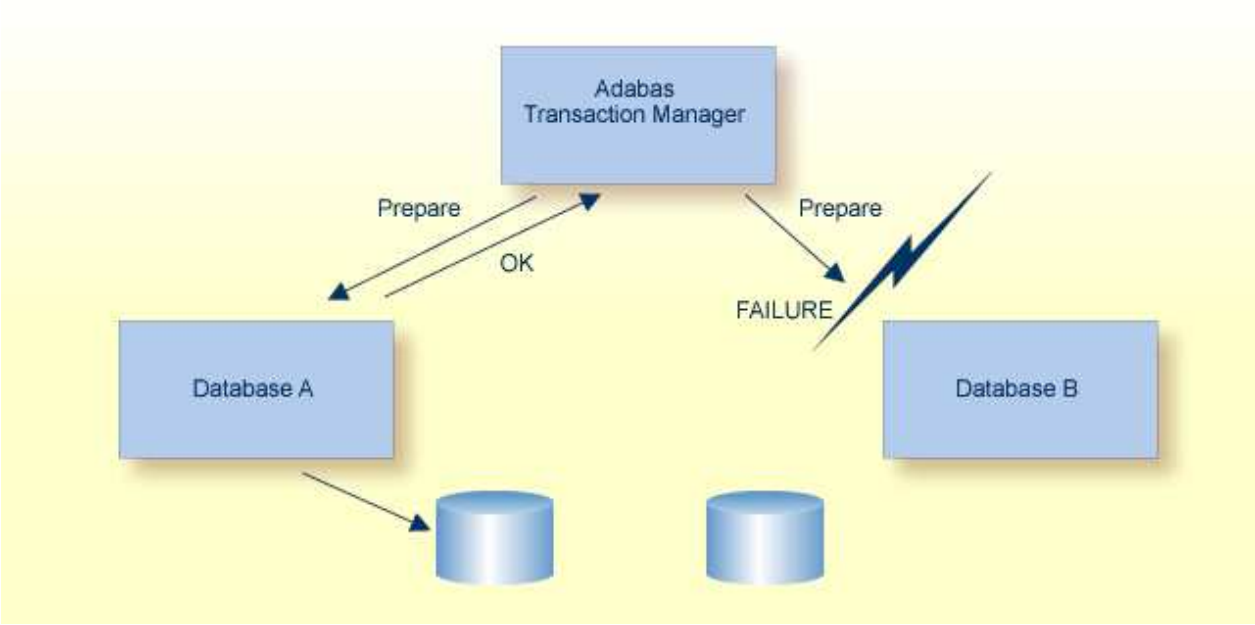
An application program processes a transfer of funds from a savings account to a checking account. The accounts are stored in different databases.

First, the application reduces the balance in the savings account record in database A and increases the balance in the checking account record in database B.

Next, it decides to commit the changes and issues an ET command. The transaction manager client proxy asks the ATM transaction manager to commit the global transaction. The transaction manager first issues a prepare request to database A and a prepare request to database B.

If both databases respond that they have prepared the transaction successfully, the transaction manager instructs each database to complete its part of the transaction by committing the changes.

However, suppose database B is unable to complete the prepare operation for some reason.



The transaction cannot be completed. The transaction manager initiates phase 2 by issuing a remove or roll back request to each database.

- Database A has prepared successfully the transaction, and is therefore able to reinstate the original balance in the savings account record using the recovery information it stored during the prepare operation.
- Database B has not prepared the transaction, so its standard roll back processing or (if the database itself failed) recovery logic reinstates the original balance in the checking account record.

Serial Mode

Serial Mode Transaction Control

In the absence of Adabas Transaction Manager and the associated software components, a global transaction must be committed by a series of ET commands, or backed out by a series of BT commands. A 3GL application must do this explicitly. Natural implements this automatically and transparently if it detects that a transaction has changed more than one database. This method of transaction control is known as serial mode. It does not provide the global transaction integrity guaranteed by two-phase commit in all circumstances.

Suppose an application is running and the ATM transaction manager is operating. If some component of the transaction management software fails; for example, if the ATM transaction manager becomes unavailable, the application receives a response code indicating that a problem exists and that global

transaction integrity can no longer be guaranteed. Once such an error condition has been reported to the application, the transaction manager client proxy component can switch the user automatically to serial mode, if set up to do so.

The application may react to the error condition (by restarting, for example), then continue to operate exactly as it would if the transaction management software were not installed. This means that, in case of a critical failure of the transaction management software, applications can continue to execute without any serious interruption, but also without the guarantee of all-or-none integrity for global transactions. Operating in serial mode carries no risk if only a single resource manager is changed in each transaction, even if the transaction is controlled by an external coordinator.

At every suitable opportunity, the transaction management client proxy component attempts to switch the user back from serial mode to distributed mode, but does not report an error to the application if it fails to do so. When the switch is successful, the application continues to execute, once again with guaranteed global transaction integrity.

Alternatively, the transaction management client proxy may be set up to regard such a failure as critical so that applications are not allowed to continue without the guarantee of global transaction integrity. In this case, transaction management returns a response code to the application indicating the nature of the failure. The application cannot continue normal processing until the fault is corrected.

The use of serial mode is controlled using the client control EmergencySerial ETCommands.

Serial Mode and ET Data

Depending on configuration options, it might be important for the ATM transaction manager to be active whenever ET data is written or read.

If the ATM transaction manager is inactive and a user is switched to serial mode, ET data requests are directed to the database specified in the Adabas control block and not to ATM transaction manager. This could result in

- incorrect ET data being read.
- ET data being written to a database from which ATM transaction manager is not able to retrieve it.
- incorrect results when ATM transaction manager attempts to recover a transaction on behalf of the user.

For these reasons, it is strongly recommended that the client control EmergencySerial ETCommands is set to NO if the ATM transaction manager is to manage ET data (see also the parameter setting TMETDATA=ATM).

If an ETID is used to read or write ET data through a link module that has an active transaction management client proxy component and client runtime control ATM=ON, then for the same reasons it is important that the ETID always use a link module with an active client proxy component and with the client runtime control setting ATM=ON.

Conversely, if the ETID uses a link module with no ATM component available, or with ATM=OFF, it must always use a link module with no transaction management processing component, or with ATM=OFF.

Interfacing with Adabas Applications

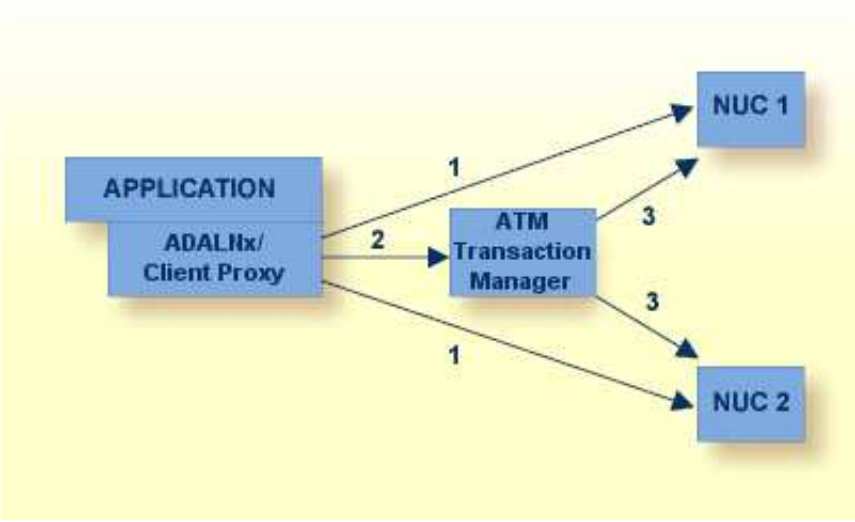
- Overview
- Adabas Transactional Commands

Overview

New or existing Adabas application systems written in Natural or a 3GL participate in two-phase commit processing transparently when the ATM client proxy component is available to the Adabas link module for each supported client environment (batch, Com-plete, CICS, UTM). The client proxy functions as a transparent application stub.

The client proxy invokes ATM transaction manager on behalf of the application in two-phase commit processing and responds to the application according to the result. It transmits information to the ATM transaction manager about the start, commit, and roll back points of a global transaction as they occur.

The client proxy keeps the ATM transaction manager informed of changes in the distribution scope of the global transaction so that it can correctly manage commit, roll back, and restart processing.



1. The application communicates with its target Adabas databases in the normal way. The client proxy monitors the commands issued by the application, selecting any that are transactional and reacting appropriately.
2. Depending on the Adabas commands issued by the application, the client proxy issues appropriate requests to the transaction manager.
3. The transaction manager issues commands to the Adabas nuclei participating in the transaction in response to events such as requests from the client proxy.

Adabas Transactional Commands

Certain standard Adabas commands trigger special processing by the Adabas Transaction Manager's components. These commands are referred to as transactional because they affect or are related to the processing that ensures transaction integrity. The transactional commands are listed below together with

information about how they are handled.

- Data Modification Commands
- Termination Commands: ET and BT
- Close Commands
- Open Commands
- RE Commands
- Ending a Transaction – Application Protocol

Data Modification Commands

The data modification commands `A1,E1,N1`, and `N2` change application data in Adabas files.

The first data modification command issued to a database that is enabled for two-phase commit processing (parameter setting `DTP=RM`) by a client session at global transaction status marks the beginning of a global transaction. Subsequent data modification commands issued to such databases may bring other databases into the scope of the global transaction.

Data modification commands issued to databases with the parameter setting `DTP=NO` have no effect on global transaction logic; they execute in the normal way and cause no special processing by Adabas Transaction Manager. Even though these changes cannot be fully synchronised with changes made to `DTP=RM` databases, Adabas Transaction Manager will do its best to ensure that they are committed or backed out along with the client's global transaction, if there is one. Changes made to `DTP=NO` databases are committed or backed out:

- when Adabas Transaction Manager commits or backs out the client's current global transaction, if there is one, or
- when the client issues `ET,BT,CL`, or `OP` to those databases, or
- when the respective Adabas nuclei reverse them using autobackout operations.

If the client is operating in serial mode following some component failure, Adabas Transaction Manager takes no part in processing data modification commands; they are simply passed to the appropriate Adabas database(s). Even so, Adabas Transaction Manager will do its best to ensure that changes to all databases are either committed or backed out together.

Termination Commands: ET and BT

Natural or 3GL applications currently terminate a global transaction by issuing a sequence of `ET` or `BT` commands, targeting each changed database in turn:

- as a result of the first such command, Adabas Transaction Manager commits (`ET`) or backs out (`BT`) the current global transaction.
- subsequent `ET` or `BT` commands are directed to the target database only, as there is no longer a global transaction in progress.

Note:
A 3GL application that manages its own subtransactions (that is, it commits changes on one database while leaving changes uncommitted on another) must run without an Adabas Transaction Manager client proxy, or with client runtime control `ATM=OFF`, and cannot therefore make use of Adabas Transaction Manager.

Processing Logic

A termination command issued by a client who is operating in distributed transaction processing mode always triggers a commit (ET) or back out (BT) operation, even if the command is directed at a database running with the parameter setting DTP=NO.

- If a global transaction is open for the client, Adabas Transaction Manager attempts to commit it (or back it out) using the two-phase commit protocol if necessary, regardless of the database that is the target of the command.
- If the command is directed at a database that has not been changed by the current global transaction or a database that is running with DTP=NO, the command is passed to the specified target after the global transaction has been committed (or backed out).
- An equivalent termination command is sent to any other changed database that runs with DTP=NO where there are uncommitted changes or held records.
- If there is no global transaction open, the command is simply passed to the specified target.

  If the client is operating in serial mode following some component failure, the ATM transaction manager takes no part in processing ET or BT commands. However, if the Adabas Transaction Manager client proxy detects an ET or BT command after one or more databases have been changed, it will issue similar commands to all changed databases in sequence.

With ET Data

If an ET command with ET data is issued by a client in distributed transaction processing mode, the processing just described occurs, but the ET data is also stored according to the value of the Adabas Transaction Manager ADARUN parameter TMETDATA:

- If TMETDATA=ATM, ET data is stored in the ATM transaction manager's database.
- If TMETDATA=TARGETS and a global transaction is open for the client, ET data specified on the ET command is stored in all databases with DTP=RM involved in the transaction. The ET data is stored in the database specified in the ET command whether it has DTP=RM or not. If it has DTP=NO, storage of the ET data in this database is not synchronized with the completion of the global transaction, but occurs after the global transaction has been committed.

  If TMETDATA=TARGETS and no global transaction is open for the user, ET data is stored only in the database specified in the ET command.

  Note:
  Any ET commands generated by Adabas Transaction Manager client proxy when it detects an incomplete sequence of ET commands are issued without ET data.

  The ET data is committed during the two-phase commit process for the global transaction; if the global transaction is backed out, the user's ET data also reverts to its previous state (in all affected databases when TMETDATA=TARGETS).

Extended Hold Processing

PET and M command options allow a program to commit (ET) or back out (BT) a transaction while leaving some records in hold status. Adabas Transaction Manager's transparency feature supports these options when the ExtendedHold client runtime control is set. Extended hold processing proceeds as follows:

- When Adabas Transaction Manager commits or backs out the current global transaction as a result of the first ET or BT command in the sequence, it has no knowledge of any P or M options that are to be applied to the databases involved in the transaction except those specified on the first command.
- Adabas Transaction Manager therefore assumes that all held records are to remain in hold status, except in the database that is the target for the first ET/BT command. It honors the P or M option specified in the first command, but ensures that the other target databases do not release their held records during the two-phase commit process. Thus, the user's global transaction is committed or backed out as requested, but at this stage some of the transaction's held records may still be in hold status.

  As each subsequent command in the sequence is issued, Adabas Transaction Manager's client proxy passes the command on to its target, ensuring that any P or M option is honored. Adabas Transaction Manager expects the application to issue a complete sequence of ET or BT commands – one for each changed database. However, if the proxy detects a non-transactional command from the application before the sequence is complete, the sequence ends and the proxy generates a command for each affected database that has not received an ET or BT during the sequence, indicating that all the held records are to be released.

If a user has been switched to serial mode because of the failure of some component, and extended hold is active, the client proxy will continue to process ET and BT commands as described above. The only difference is that any changes will be secured by a series of commands, without the synchronization that would normally be guaranteed by the two-phase commit protocol used by the ATM transaction manager.

If the extended hold option is not active (the default setting), any ET or BT command will cause all the user's held ISNs to be released in databases where changes have been made, except possibly in the database that is the target of the command; for this database, any P or M option present in the command will be honored.

Note:
Extended hold processing does not occur when an unsolicited syncpoint occurs. See the client control TransactionControl for more information.

Using ET Data IDs (ETIDs)

It is possible for a client to use an ETID for some database sessions, and concurrently to use other database sessions with no ETID. It is also possible for a client to use different ETIDs concurrently in different database sessions; however, Software AG strongly recommends that you avoid this practice.

Close Commands

Since a close command implies end-of-transaction, every CL command triggers the same processing as for an ET command, except that:

- the user is also closed in the target database;
- if the target database is at ET status, the current global transaction is not affected.

Open Commands

If an OP command is issued by a client who has no global transaction in process, and the user has no uncommitted changes in the target database, the command is simply passed to the indicated target database.

An OP command sent to a database in which the client has uncommitted changes or held records will cause the client's uncommitted changes on all databases to be backed out and all held records to be released, regardless of their distributed transaction processing parameter settings.

An OP command sent to a database in which the client has no uncommitted changes or held records will simply be passed to its target without affecting the user's global transaction status. This logic allows "open on demand" processing without interference in global transaction processing.

If the OP command specifies that ET data is to be read, the above processing occurs. If processing is successful, the user's ET data is returned to the calling program. This applies even if the target database has the parameter setting DTP=NO. The ET data is read either from the ATM transaction manager's database or, if the transaction manager is running with the parameter setting TMETDATA=TARGETS, from the database indicated by the OP command.

RE Commands

If a user issues an RE command while operating in serial mode, the command is simply directed to the indicated target database without any interference by Adabas Transaction Manager.

If an RE command is issued in distributed transaction processing mode, the required ET data is obtained either from the ATM transaction manager's database or, if the transaction manager is running with the parameter setting TMETDATA=TARGETS, from the database indicated by the RE command.

Ending a Transaction – Application Protocol

Adabas Transaction Manager acts to end a distributed transaction across all affected databases, with full transaction integrity, as soon as the application session issues an ET or BT command. Nevertheless, Adabas Transaction Manager expects an application to be well-formed in the way it pursues transaction outcome. Specifically, where multiple databases are modified, the application is expected to pursue transaction outcome on all the changed databases, in series. For example, if two databases are modified, Adabas Transaction Manager expects to see two ET commands (one to each database) to cause a positive transaction outcome; or two BT commands (one to each database) for a negative transaction outcome. If an application tries to issue a transactional command without having issued the expected ET/BT commands to all databases changed by the previous transaction, response code 240, sub-code 496 will be returned.

Pending Response Codes

Sometimes a transaction is terminated in such a way that the owner should receive a non-zero response code, but in circumstances in which it is not possible to return a response code; for example, the transaction manager backs the transaction out because its global transaction time limit has been exceeded. When this happens, the manager stores details of the pending response code in a list, and returns it to the

client at the first possible opportunity. Pending response codes can be listed and displayed using ATM's Online Services application.

As with standard usage of Adabas, the time for which a pending response code is preserved depends on whether ETIDs are used.

If the transaction which caused the pending response code involved no RM database sessions with ETIDs, the pending response code is discarded as soon as any of the following conditions has been satisfied:

- The response code has been returned to the client.
- The client is known to have disappeared.
- All DTP=RM databases that took part in the transaction have been restarted.
- The ATM manager terminates.

If the transaction which caused the pending response code involved one or more RM database sessions with ETIDs, the pending response code is discarded as soon as any of the following conditions has been satisfied:

- The response code has been returned to the client.
- The client session is known to have disappeared.
- All DTP=RM databases that took part in the transaction, and for which an ETID was in use, have been restarted.

Note:
In this case a pending response code will survive a restart of the ATM manager.

Adabas Transaction Manager Support for ET Data

ET data is a special type of application data. Changes to ET data take effect if and only if a transaction completes successfully.

Adabas Transaction Manager supports ET data with global transactions. If the global transaction completes successfully, the new ET data is stored; otherwise, the existing ET data remains unchanged.

If an application issues an ET command with ET data, on a session for which no valid ETID has been provided, ATM, like Adabas, will not store the ET data persistently. Instead, it will proceed as follows:

- If the target database of the ET command is being managed using the two-phase protocol, the ET data will be ignored, but ATM will not draw attention to this fact by means of a non-zero response code;
- Otherwise, the ET data will be passed to the database which the application specified on the ET command, and to no other database.

If ET data is stored by an ET or CL command on a session which has an ETID, its default location is determined by the Adabas Transaction Manager ADARUN parameter TMETDATA:

- When considered part of the global transaction, ET data is stored in and read from the ATM transaction manager's database (TMETDATA=ATM). When an application wants to read ET data, the transaction manager satisfies the request from its own database, without regard to the database specified in the Adabas command.
- When considered part of a particular database, ET data is stored in every database affected by the global transaction (TMETDATA=TARGETS). An application must issue a suitable command to an appropriate database in order to read ET data.

The default location for ET data can be overridden by means of the client runtime control Application controls ET data . If this control is set to YES, the ET data is stored only in the database to which the command was issued.

It is strongly recommended that a client should not use more than one ETID concurrently in different database sessions. If this recommendation is followed, there can be no confusion about which ETID should be used when ET data is to be stored or read. If the recommendation is not followed, and the ADARUN parameter TMETDATA=ATM is specified, and is not overridden by the client runtime control Application controls ET data, ATM will assume that an ET data operation should use the ETID that is currently associated with the database to which the ET, CL, OP or RE command has been issued.

For information about:

- controlling the storage of ET data and synchronizing storage across transaction coordinators, see section ET Data Storage and the TMETDATA parameter.
- establishing the current ET data of your applications in the ATM transaction manager's database before they execute, see Copy ET Data.
- how ET commands with ET data are handled in a distributed environment, see section Adabas Transactional Commands.
- problems handling ET data in serial processing mode, see section Serial Mode.
- handling ET data in application environments where dynamic transaction routing can take place, see section Dynamic Transaction Routing .

Adabas Transaction Manager Support for Triggers

Triggers can execute global transactions under the control of Adabas Transaction Manager. Refer to the documentation for the Adabas System Coordinator to find out how to configure the System Coordinator for a trigger environment.

If your application causes participating triggers to be fired, and you require a global transaction to include changes made by both the application program and the participating trigger, you must ensure that the very first change command (store, delete or update) of the global transaction is issued by the application program, not by the trigger.

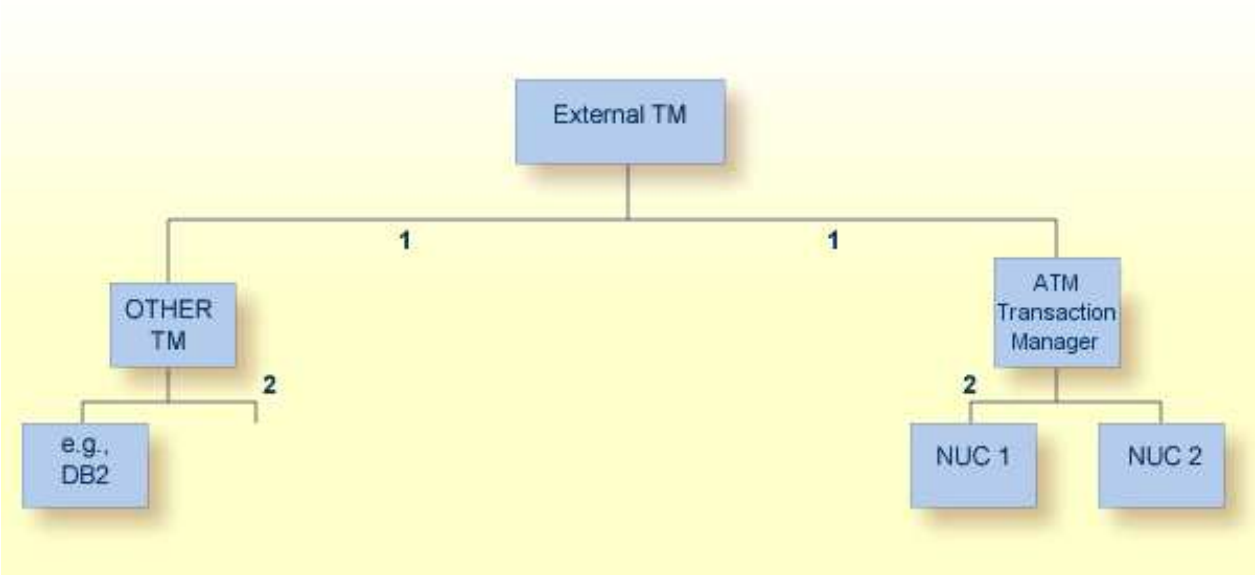
Interfacing with Other DBMSs

- Using External Transaction Managers
- Using the ATM CICS RMI and the CICS Syncpoint Manager
- Using RRMS
- Dynamic Transaction Routing

Using External Transaction Managers

Adabas Transaction Manager also provides configurable facilities (APIs) to respond to directions from external transaction managers such as

- IBM’s Recoverable Resource Management Services (RRMS); or
- the CICS Syncpoint Manager



1. An external transaction manager is responsible for the two-phase commit process. When instructed by the application (or some other agent), it directs the subordinate transaction managers that are involved in a transaction to prepare the transaction and, if all prepares are successful, to commit the transaction.
2. Each subordinate transaction manager is responsible for implementing each phase of the two-phase commit process separately, as directed. For example, the ATM transaction manager issues a prepare to each Adabas nucleus participating in the transaction, then takes no further action for the transaction until it receives the instruction to commit or roll back from the external transaction manager.

When participating in global transactions with non-Adabas DBMSs, the Adabas Transaction Manager components jointly handle all communication with the external transaction manager. Using the interfaces provided, an appropriate component of Adabas Transaction Manager registers with the external transaction manager and agrees to use the asynchronous exits that it drives when two-phase commit

processing is required. A single instance of Adabas Transaction Manager thus responds to the external transaction manager on behalf of all the Adabas databases within the network. For sites with a large number of databases, this provides an additional efficiency bonus.

Using the ATM CICS RMI and the CICS Syncpoint Manager

Adabas databases defined as resource managers according to the requirements of the Adabas Transaction Manager CICS Resource Manager Interface (RMI) are directly coordinated by a locally executing ATM transaction manager which, in turn, acts as a resource manager coordinated by the external CICS Syncpoint Manager. Other resource managers coordinated by the CICS Syncpoint Manager can include other Adabas Transaction Managers and/or DBMSs.

Adabas databases in a CICS environment can participate along with any combination of other DBMSs in global transactions that are coordinated by the CICS Syncpoint Manager, which drives the two-phase commit protocol. All the information necessary to recover to a consistent state after a failure is collected in CICS logs and the logs of the subordinate resource managers.

Adabas Version 7.1 and above implements RMI through the Adabas task-related user exit (TRUE). From Version 7.4, this facility has been available on VSE/ESA systems, as well as z/OS systems.

A transaction that is coordinated through the CICS/RMI can change Adabas databases on local or remote systems. Syncpoints are coordinated across all affected resource managers and Adabas databases if

- the Adabas databases run with parameter setting `DTP=RM`;
- each Adabas database runs under the control of an ATM transaction manager in the same system; and
- the affected systems are connected by Entire Net-Work.

Databases running with the parameter setting `DTP=NO` can also be changed by the same transaction; Adabas Transaction Manager will issue `ET` commands to commit the changes, but these will not be synchronized with the two-phase commit process.

Example

A global transaction changes some Adabas databases, a DB2 database, and some VSAM data. The CICS Syncpoint Manager controls the two-phase commit process across all DBMSs. The application’s decision to commit might be signaled by

- an `ET` command;
- an `EXEC CICS SYNCPOINT` command; or
- CICS task end.

In each case, the CICS Syncpoint Manager requests each resource manager (ATM, DB2, and VSAM) to prepare and then commit (assuming all prepares are successful) its part of the global transaction.

Because Adabas Transaction Manager is able to coordinate global transactions across operating system images, the support for CICS/RMI also covers transactions that change Adabas databases across system boundaries.

Using CICS RMI with APPC Applications

The Adabas Transaction Manager CICS RMI implementation makes it possible for server programs, executing under CICS, to make Adabas changes under the transactional control of a remote CICS region or some other agent. The most common way of achieving this is through the use of Advanced Peer-to-Peer Communication (APPC).

The programmer of an APPC application must ensure that the protocol is followed correctly. Syncpoints may be executed only when the program has the correct conversation state, otherwise abends such as ASP2 can occur.

In such an environment, a `SYNCPOINT ROLLBACK` operation changes the conversation state of all participants. In particular, a `BT` command should not be followed by an `ET` command in an APPC application, otherwise ASP2 abends are likely.

If you plan to use the CICS RMI implementation with APPC applications, please refer to the IBM documentation and ensure that your applications follow the correct protocol. Remember to consider the syncpoints that ATM generates when `ET`, `BT` and `CL` commands are issued.

Using CICS RMI in Serial Mode

Serial mode execution allows applications to continue processing without interruption, if the ATM transaction manager should be unavailable for a period, but without the guarantee of transaction integrity in the case of system failure. If a CICS client who is running under the RMI is switched to serial mode, the Adabas Transaction Manager client proxy will ensure that any CICS syncpoint (commit or roll back) is propagated to all changed Adabas databases, through the use of a series of `ET` or `BT` commands. This does not prevent the possibility of a mixed result (some databases committed and others backed out) in the case of a system or component failure, but it does allow CICS applications to continue running until the transaction manager can be restarted.

Using RRMS

Using RRMS in z/OS Environments

In z/OS environments, the IBM Recoverable Resource Management Services (RRMS) makes it possible for any combination of resource managers (typically DBMSs) to participate in global transactions that are coordinated by Resource Recovery Services (RRS), a component of RRMS that drives the two-phase commit protocol. Its logs and the logs of the subordinate resource managers provide all the information needed to recover to a consistent state after a failure.

To participate in this process Adabas, like any other DBMS, must be implemented as an RRMS-compliant resource manager. This is possible with Adabas Version 7.4 and above, and Adabas Transaction Manager. The locally executing ATM transaction manager participates as an RRMS resource manager on behalf of all Adabas databases on mainframe systems, both local and remote.

For example, suppose a global transaction changes several Adabas databases and a DB2 database. RRMS controls the two-phase commit process by driving exit routines provided by DB2 and Adabas Transaction Manager. The application’s decision to commit can be signaled by

- an `ET` command;

- a system call to RRMS from the application itself; or
- a system call to RRMS made by some other agent on behalf of the application.

In each case, RRMS executes the prepare exit routines of each resource manager (ATM and DB2) and then, if all prepares are successful, their commit exit routines.

Because Adabas Transaction Manager is able to coordinate global transactions across operating system images, its RRMS support also handles transactions that change Adabas data across system boundaries.

RRMS support is provided for single-user, single-TCB batch applications, and for applications running under IMS TM.

Using RRMS with IMS TM

If your applications run under IMS TM, and IMS allows its transactions to be coordinated by RRMS, Adabas Transaction Manager can ensure that their Adabas changes are committed (or backed out) in a synchronized manner, under the control of RRMS. An IMS commit syncpoint causes RRMS to carry out a commit operation for all changed resources that are managed by RRMS-enabled resource managers. An IMS rollback syncpoint causes all changes to be backed out.

The completion of an IMS message (normally this means screen I/O) causes a syncpoint to take place. In the case of successful completion, this is a commit syncpoint. Moreover, a commit syncpoint implies the completion of processing for the current message. For this reason, an `ET` or `CL` command that is issued during the processing of a message will not cause an RRMS commit syncpoint; any pending Adabas changes will be committed, but non-Adabas resources will be unaffected. Further, any held ISNs will be released, or will remain in held status, depending on the presence of P or M command options, and the setting of the Extended Hold client runtime control.

The syncpoint that takes place at message end is unsolicited, from Adabas Transaction Manager's point of view. As in the case of unsolicited syncpoints under the CICS Resource Manager Interface, this syncpoint will affect pending changes in any Adabas databases, and will cause all held ISNs to be released.

A roll back operation is signaled by an IMS rollback command, or by abnormal termination of message processing. Rollback can occur during the processing of an IMS message, and does not necessarily indicate that processing of the message has completed. Any Adabas command that causes a backout to take place will trigger an IMS rollback syncpoint; this causes an RRMS roll back for all changed resources that are managed by an RRMS-enabled resource.

Dynamic Transaction Routing

Using the Adabas System Coordinator, Adabas Transaction Manager can be used to provide support for distributed transactions in environments where dynamic transaction routing can take place. This includes application environments such as CICS MRO and IMS TM. Adabas Transaction Manager will recognize a user who has been dynamically migrated from another address space, or even from another operating system image, and will allow the user to continue processing. If your systems allow a user to be relocated dynamically to a different system you should set the ADARUN parameter `TMETDATA=TARGETS` for your ATM transaction managers. The alternative setting, `TMETDATA=ATM`, relies on each client having a consistent (unchanging) local ATM transaction manager for the duration of the client session.

If your systems use ETIDs, and allow clients to be dynamically relocated between systems, you must take care to ensure that each ETID is unique across all the systems.