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
- Open Distributed Transaction Processing
- Processing Modes
- Interfacing with Adabas Applications
- ET data support
- Triggers support
- Dynamic Transaction Routing support

The Role of the Adabas Transaction Manager

Adabas Transaction Manager provides distributed transaction support for Adabas systems; coordinating changes to Adabas databases in a seamless, integrated way. 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 facilities of the organization.

Adabas customers have significant investments in existing application systems. Adabas Transaction Manager enables these systems to participate in distributed transaction processing transparently without having to incur significant further investment to make them compliant.

Open Distributed Transaction Processing

- General
- Open distributed transactions
- Support for Distributed Transaction Processing in Adabas

General

Distributed Transaction Processing is addressed by the well-known industry standard, Open DTP. The standard describes how an application is able to modify multiple data sources in a single transaction - guaranteed to be safe so all the modifications are wholly applied (committed) or wholly undone (backed out). The DTP standard includes the following (in brief):

- the application (AP) which executes the logic to do distributed transactions;

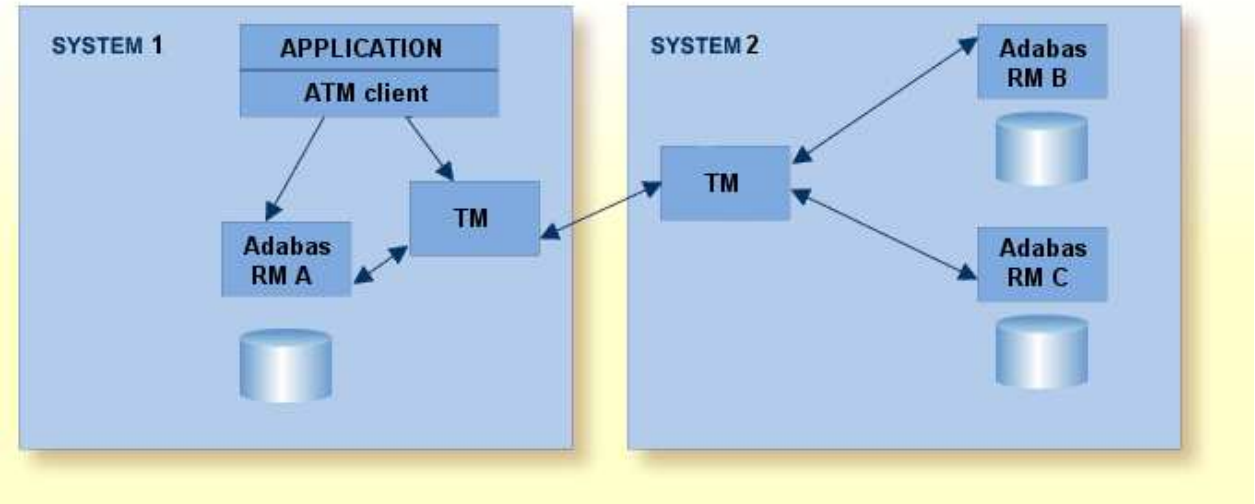
  one or more resource managers (RM). RM is the owner of a data source modified by AP. Usually RM equates to a DBMS (such as Adabas), there are other types of RM such as those that are able to manage transactional middleware messages for example;

- the transaction manager (TM) coordinates the transaction activities of RMs and APs within a single operating system image; and

- the communication resource manager (CRM) coordinates inter-system TM activities when a distributed transaction spans more than one operating system image.

By adopting Adabas Transaction Manager, DTP compliance happens transparently without having to change the logic in your application.

Note:
Distributed transactions that span multiple operating system images require use of Entire Net-Work.



Open distributed transactions

The Adabas Transaction Manager architecture is compliant with the Open DTP standard. This means Adabas transactions can be seamlessly synchronized with the products from other venders such as DB2, VSAM, MQ Series, etc. In this open scenario Adabas Transaction Manager ensures DTP compliance for the Adabas domain in collaboration with other vendor domains.

The most common DTP compliance infrastructure from another vendor is IBM’s CICS/RMI. Adabas Transaction Manager can be configured to inter-operate with CICS/RMI so that Adabas transactions inside CICS are coordinated along with those of DB2, VSAM, etc that take place in the same CICS task as the Adabas application.

Adabas Transaction Manager can also be configured to inter-operate with IBM’s Recoverable Resource Management Services (RRMS) to address; a) batch job, b) Com-Plete, and c) IMS TM support, in addition to, and separate from, CICS.

Support for Distributed Transaction Processing in Adabas

Adabas incorporates nucleus functions to support the execution of distributed transaction processing transparent to existing Adabas application systems (including 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

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

Adabas Transaction Manager provides distributed transaction support for Adabas Cluster Services and Adabas Parallel Services databases in addition to standard edition Adabas.

Processing Modes

- Distributed Transaction Mode
- Continuous operation

Distributed Transaction Mode

DTP is the default mode when Adabas Transaction Manager is used. DTP mode uses two-phase commit protocol as necessary internally when the AP requests commit (commit is usually requested by issuing an ET command). The internal phases are briefly described here:

- Phase 1: Prepare
- Phase 2: Commit or Undo
- Examples of commit and undo transaction outcomes

Phase 1: Prepare

During phase 1, each modified Adabas RM is directed to prepare its changes by the TM. This means each Adabas RM and its associated logs must reach a state where the changes can either be applied or undone. Once the Adabas RM has prepared the transaction it is in a position to either apply or undo – according to direction by the TM during phase 2.

When all Adabas RMs achieve prepared state, the TM opts to commit (apply) them all during phase 2. However, if one Adabas RM fails to prepare, the TM opts to tell all participating Adabas RMs to undo their portion of the distributed transaction.

Phase 2: Commit or Undo

Based upon the result of the preparation phase, the TM instructs each Adabas RM to either commit or undo.

Examples of commit and undo transaction outcomes

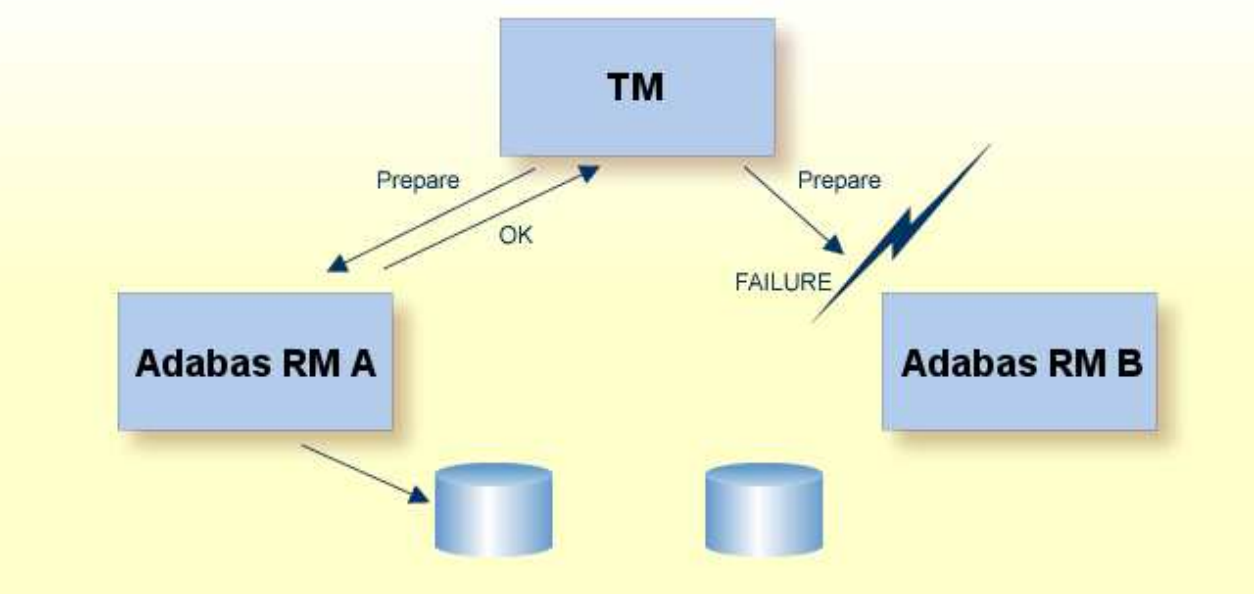
The AP processes a transfer of funds from a savings account to a checking account. The accounts are stored in different Adabas RMs (databases).

First, the AP reduces the balance in the savings account record in Adabas RM A and increases the balance in the checking account record in Adabas RM B.

Next, it decides to commit the changes. The TM first issues a prepare request to Adabas RM A and then a prepare request to Adabas RM B.

If both Adabas RMs respond that they have prepared the transaction successfully, the TM instructs each Adabas RM to complete its part of the transaction by committing the changes so the distributed transaction is now wholly applied.

However, suppose Adabas RM B is unable to complete the prepare operation for some reason – at this point Adabas RM A is waiting to commit but Adabas RM B has a problem preventing the transaction outcome being committed in all Adabas RMs, as shown here:



The transaction cannot be wholly applied because one of the two Adabas RMs has suffered an outage. The TM initiates phase 2 by issuing undo requests to each Adabas RM so the distributed transaction is wholly undone:

- Adabas RM 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.
- Adabas RM 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.

Continuous operation

In the past, many Adabas systems modified multiple RMs without using Adabas Transaction Manager thereby risking inconsistent results across RMs (refer to the previous example of commit and undo transaction outcomes). This is one obvious reason for adopting Adabas Transaction Manager. In addition, the need for distributed transactions increases all the time based upon ever growing data volumes in turn pushing the need for distributing data across ever more databases and computers. Adabas Transaction Manager allows you to distribute data in any way you wish, in the knowledge that your transactions are safe.

However, it is inevitable that from time to time transient unplanned outage of one or more components of Adabas Transaction Manager may occur. Adabas Transaction Manager acknowledges this inevitability by providing various options for the level of continuous operation you wish to have and the level of error tolerance during these transient component outages.

For example, one business may take the position that continuous (uninterrupted) operation is more important than transient component outages causing errors for all user transactions during the outage. There may be a large number of transactions happening which in turn means a large number of potential error reports can occur very quickly – which in itself can cause problems in support centers. Alternatively, another business may take the position that it is better to get errors, no matter how many arise during these periods.

Significantly, transient outage of Adabas Transaction Manager does not automatically result in inconsistency. First, Adabas itself is extremely reliable so inconsistencies are rarely caused by failure in Adabas. However, whole system outage in the middle of an application issuing ET to two Adabas RMs would result in inconsistency if Adabas Transaction Manager is not used. This is clearly a rare situation but it is a risk that has to be considered, which is why Adabas Transaction Manager exists.

Adabas Transaction Manager will make sure inconsistencies cannot occur during transient outage of its own components simply by refusing to allow distributed transactions. That’s the simple case. But as already stated the prevalence of numerous error reports can be disruptive – unnecessarily so since the error reports indicate only a risk of error, not that error has really happened. It is a question of whether the disruption for potential error rather than real error is worth it. This is an operational decision.

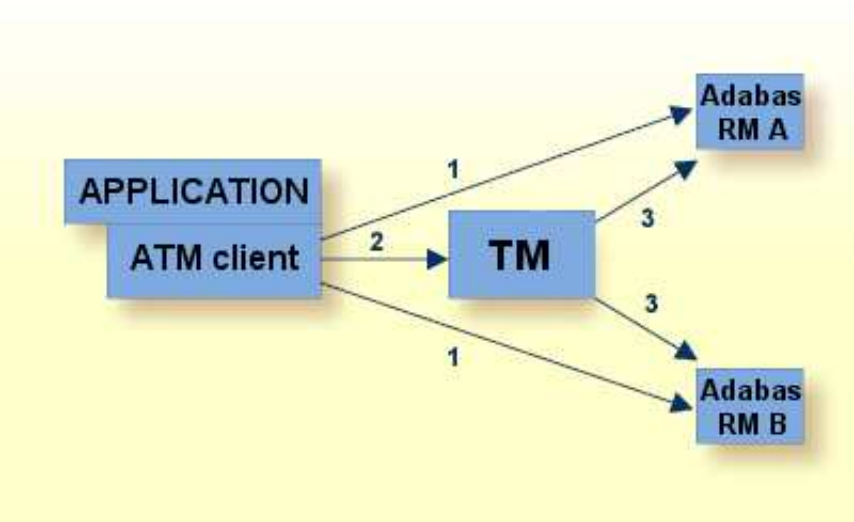
Adabas Transaction Manager can be configured to dynamically revert to issuing a series of independent ET to all changed Adabas RMs without full distributed integrity during outage…essentially meaning the systems continue to run uninterrupted in non-DTP mode with the small risk that inconsistencies could potentially occur if, for example a whole system crash occurs between two ET commands for the same user. Conversely, the configuration can be set to make sure this contingency mode does not happen so that all distributed transactions during transient outage are rejected with error responses.

The choice is based upon risk management versus the levels of interruption. If the risk of inconsistency is very low and the period of transient outage is small then one company may choose to accept the risk and allow uninterrupted continuous operation in the knowledge that full DTP mode will resume automatically as soon as the outage ends. Alternatively, you may choose to withstand the interruptions to transactions for the period it takes to rectify the transient outage.

You make the choice for your company by *Continuous operation mode*.

Interfacing with Adabas Applications

New or existing Adabas application systems written in Natural and/or 3GL are coordinated transparently when Adabas Transaction Manager is introduced; no changes are required to the vast majority of applications. The ATM client (see the diagram below) interacts with the TM and Adabas RMs automatically to provide transaction management for your Adabas applications according to the configuration you set.



1. The AP communicates with its target Adabas databases in the normal way. The ATM client 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 ATM client issues appropriate requests to the TM.
3. The TM issues commands to the Adabas RMs participating in the transaction.

ET data support

Adabas Transaction Manager can be configured to support the use of ET data in either of two ways:

- ET data are stored by Adabas Transaction Manager in its own internal recovery file
- ET data are stored where the application system dictates

Triggers support

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.

Dynamic Transaction Routing support

Single system

TP systems support dynamic transaction routing within a single system image such as CICS (MRO), IMS TM and UTM. These systems provide greater load balancing and more resilience by running multiple components in parallel.

Adabas Transaction Manager supports dynamic transaction routing in these systems so as a client session moves from TP component to component under load balancing conditions transaction management continues seamlessly.

Multiple systems

CICS/PLEX supports dynamic transaction routing across multiple system images. This provides even greater load balancing and even more resilience by running multiple components in parallel across systems so that outage of whole system images can be survived and load can be spread to an even greater degree.

Adabas Transaction Manager supports dynamic transaction routing in these systems so as a client session moves from computer to another under load balancing conditions transaction management continues seamlessly.

In this configuration a TM component must be present in each system image and they must be configured to run within the same Adabas System Coordinator group. These TM collaborate to enable transaction management to move around the SYSPLEX on demand in reaction to the TP load balancing and failover decisions that are made dynamically.