Detection of ill-formed distributed transactions

Adabas Transaction Manager applies distributed transaction integrity as soon as the first ET of a series is encountered. This suggests the subsequent ET commands of the series are not needed but that is incorrect. Well-formed distributed transaction programming is still necessary and enforced. Here's why:

1.	Modify data in Adabas database A
2.	Modify data in Adabas database B
3.	Issue ET command to Adabas database A ● ATM transparently performs two-phase commit for all modified databases
4.	Issue BT command to Adabas database B ● **The application is trying to execute an illegal operation!**

In the example above, the application is breaking the rules of distributed transaction processing by adopting unsafe behavior. The rules of DTP stipulate the modifications for all databases in a distributed transaction are all applied or all undone; it is not legal to have a mixture of some applied and some undone.

In this situation Adabas Transaction Manager throws an error when the BT in point 4 above is encountered (response 240 sub-code 496). Here's another example, just a little more complicated:

1.	Modify data in Adabas database A
2.	Modify data in Adabas database B
3.	Issue ET command to Adabas database A ● ATM transparently performs two-phase commit for all modified databases
4.	Modify data in any database (A or B or another) ● **The application is trying to execute an illegal operation!**

In this example, the application issues only one ET command, it does not issue an ET command to the second database. Then it goes on to do other work – in this case to start doing modifications in a new transaction. This logic is breaking the rules of distributed transaction processing with unsafe behavior too. In this situation Adabas Transaction Manager throws an error when the modification in point 4 above is encountered (response 240 sub-code 496). Some people find it difficult to understand why this example breaks the rules so consider what might happen without Adabas Transaction Manager being used:

1.	Modify data in Adabas database A
2.	Modify data in Adabas database B
3.	Issue ET command to Adabas database A ● Adabas database B backs out the changes made so far due to resources shortages or some other reason.
4.	Modify more data in Adabas database B
5.	Issue ET command to Adabas database B

This is really not a good situation when Adabas Transaction Manager is not present. Here is what has happened…

1.	The modification to Adabas A is applied.
2.	The first modification to Adabas B is undone.
3.	The second modification to Adabas B is applied… ● **Only half of the modifications to Adabas B are applied.**

The only way to be sure distributed transactions are executed correctly in your business systems is to make sure there is no room for doubt by

1. adopting Adabas Transaction Manager

   and

2. allowing it to enforce well-formed distributed transaction processing behavior, as it does by default.