Introducing ETP

This chapter covers the following topics:

- ETP Concept
- Data Integrity Choices
- How ETP Operates
- ETP File Structure
- Replication Task Control
- Tailoring and Tuning ETP
- Resynchronization Modes

ETP Concept

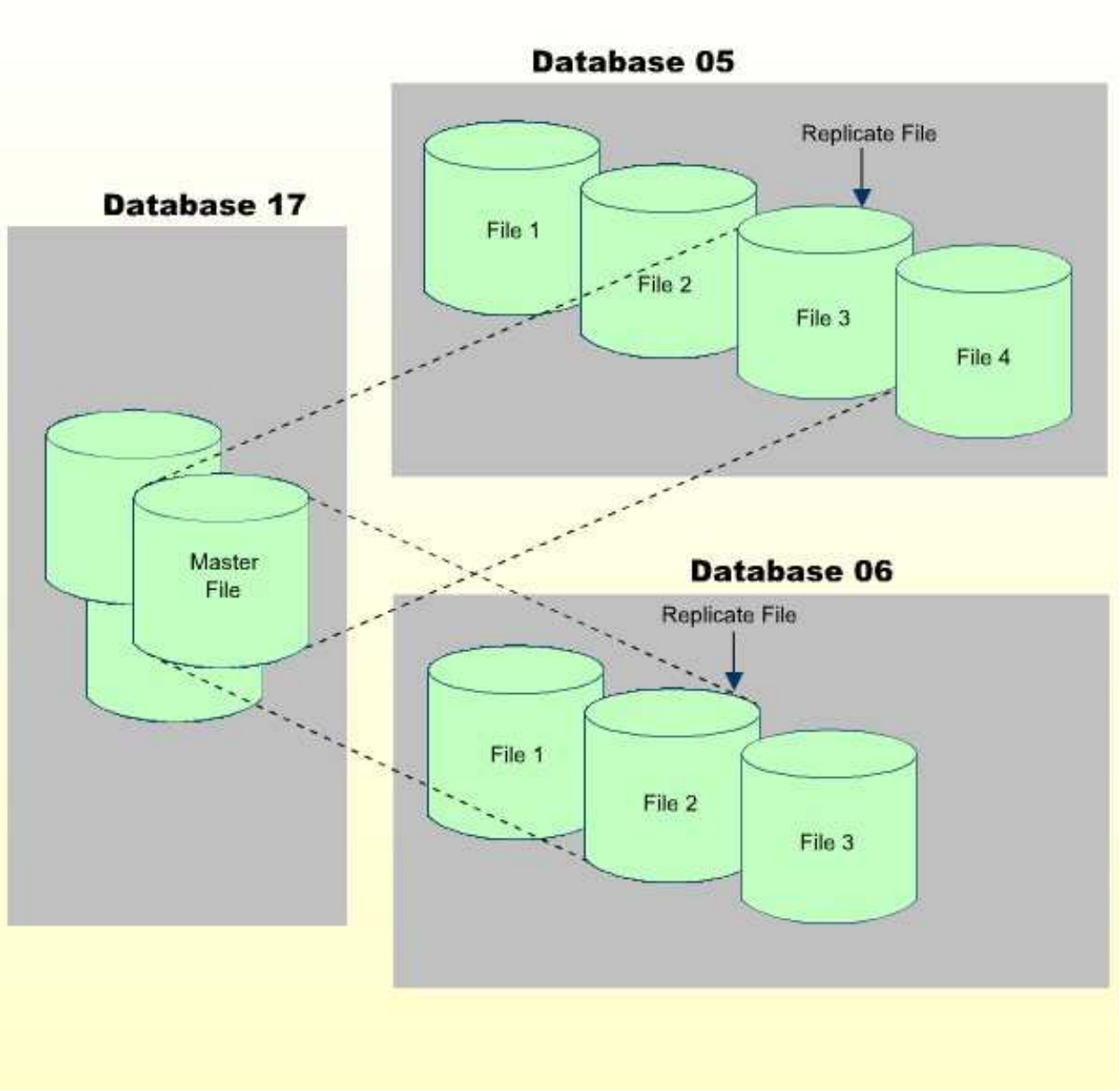
The concept of a distributed database provides much greater operating efficiency and flexibility while at the same time offering almost unlimited data capacity. Such a "networked" database structure means that the portion of the database data needed by, for example, Corporate Operations or Personnel can be located on local systems and still be available corporate-wide as part of the common database resource.

One particularly appealing feature of distributed databases is the possibility of having duplicate copies of data at those locations where the data is needed most. This concept allows duplicate copies of a data file to be located throughout the database network, yet the copies are viewed logically by users as a single file.

However, maintaining the duplicate data in each location has generally also meant a sophisticated control mechanism to ensure simultaneous integrity among all copies of the duplicate data. Although effective and proven, such "fail-safe" data integrity is often not necessary for some user environments. Software AG’s ETP offers a cost-effective alternative to these more sophisticated systems for installations that do not require absolutely simultaneous integrity in their duplicate data resource.

ETP holds the duplicate data at each location in a *replicate file* (see the figure *A Basic ETP File Structure*). The replicate file is a read-only image of data held in a *master file*. The replicate files are "refreshed" periodically if changes have been made to the master file. If and when refreshing is done and which replicate files are refreshed are the choice of the ETP administrator.

A Basic ETP File Structure



Data Integrity Choices

Conventional database design standards require replicated files to have "full integrity". This means that, whenever a logical record is available to a user program, all replicated copies of the record must contain identical and up-to-date information, regardless of where or how often the record is duplicated. Although a program performs an update transaction to only one local copy, all other copies must first be changed to reflect the transaction before another program refers to that information. It is the job of the distributed database management system (DDBMS) to ensure that all copies of the replicated file are identical and accurate at all times.

In practice, managing a replicated file to ensure data consistency in all copies requires using a concept known as "two-phase commit" logic. Here, a program that changes a replicated file receives a confirmation of the completed change only after all active copies throughout the distributed database have been changed. If one of the copies becomes unavailable, it must first be "resynchronized" with the other

copies to restore absolute database integrity. One can imagine the need for such high integrity in an airline reservation system, which has a high rate of change and possible contention.

In many database environments, however, the need for such complete integrity is not as critical. Branch offices, for example, require frequent access to item inventories to obtain item descriptions and prices for customer billing. Copies of the item inventory dataset, which is normally revised only by the main office, can be located at each branch office. The rate of change to the item inventory dataset is relatively low; each location reads the dataset frequently but changes it infrequently, if at all.

Using the Entire Transaction Propagator (ETP), Adabas database systems with a high rate of read transactions but a low update rate can have the benefits of replicated files without the complex controls needed by systems with high transaction/contention rates. Using a "master/replicate" system of control, ETP resynchronizes all replicate copies with a master copy at user-specified intervals.

Generally, most database requests are read requests. Even in systems where the requests are mostly STORE/DELETE/UPDATE (referred to generally in this text as update) operations rather than read requests, the update operations are often to only a small part of the overall distributed database. Frequently, the update requests come from a limited number of users and/or locations. Restructuring the data to "move it closer", logically speaking, to its principal "owners" and users can improve the overall system efficiency. Then, using ETP, each master file can be located where change activity is highest for that file, with replicate copies of the master file at other locations for local read-only use.

ETP also allows the replicate file copies to be "tailored". If a particular location needs only a small portion of the overall data resource, ETP allows creation of a replicate copy containing only the needed records.

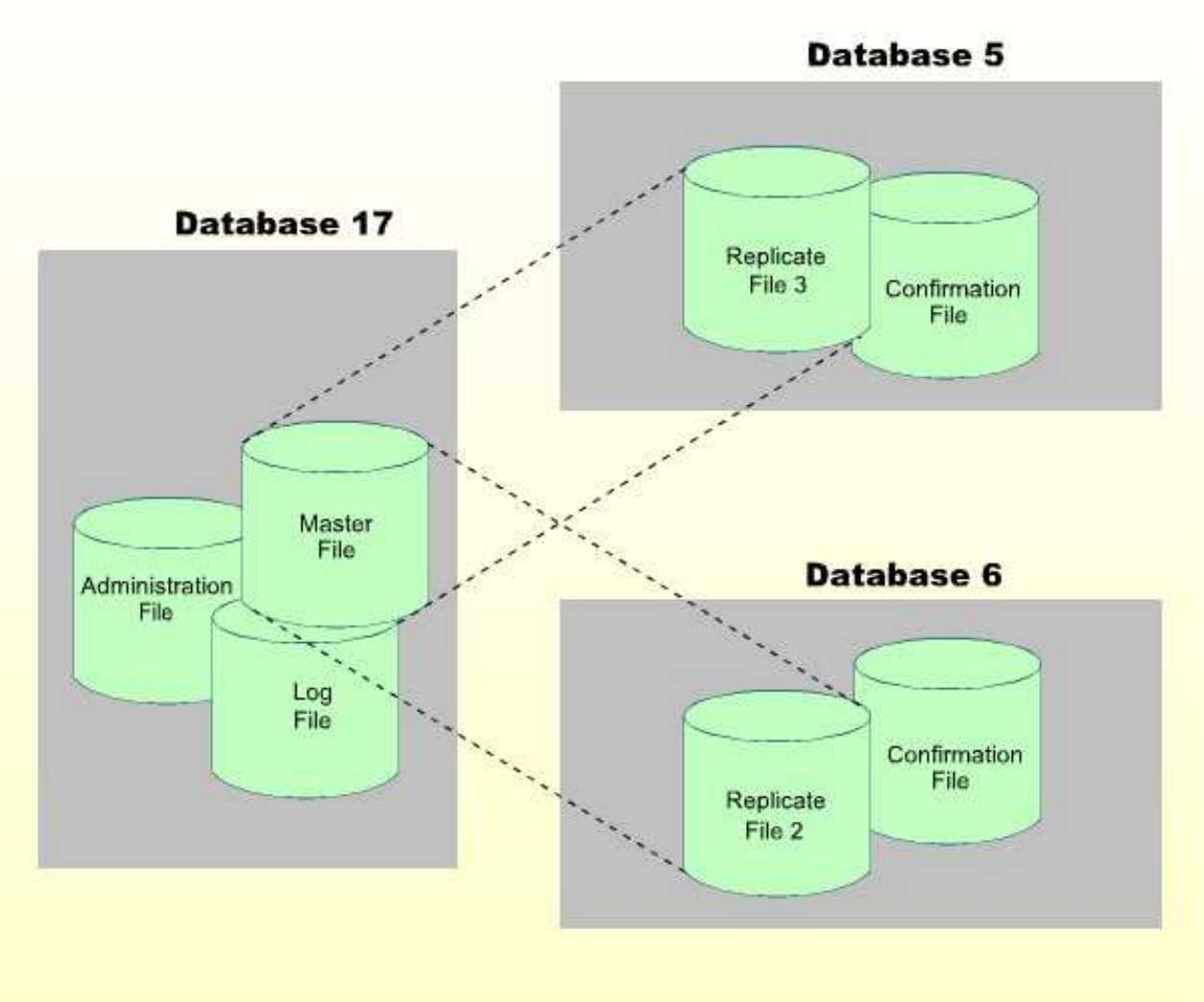
How ETP Operates

ETP provides the convenience of replicate files using a "master/replicate" concept. Here, the more frequent read requests are satisfied by the nearest replicate copy, while the occasional update requests are applied only on the master file. In addition, changes to the master copy are logged. At user-defined intervals, the replicate copy is updated with the logged changes.

In this way, the requests for reading data are resolved locally, independent of the activity in other copies and with no network activity. Programs making update requests affect only the master file, and therefore need an Adabas end-of-transaction (ET) from only the master copy before continuing their operation.

Any updates made to the master file are logged in the related log file; the database operations used for logging are part of the same transaction as the original update operation; therefore, data integrity, including the log file contents, is automatically preserved by normal Adabas safeguards.

As for any Adabas file, incomplete or otherwise unsuccessful updates transactions are restored or removed ("backed out") by the normal Adabas restart/recovery facility.



A Basic ETP Replicate File with Support Files

ETP File Structure

To manage the master file and replicate file copies, other "background" files support the ETP environment as shown in the figure *A Basic ETP Replicate File with Support Files*, above.

Each master file has a *log file* for recording update transactions. All updates are recorded in the log file as soon as they are applied to the master file. The log file, which must be on the same database as the master file, can serve all master files on a database, is the source for update information when ETP resynchronizes replicate files with their master file.

Whenever replicate files are resynchronized, the replicate files are updated and the resynchronization is recorded in a *confirmation file*. As with the master file's log file, a confirmation file must be located on the same database as the related replicate files, and can serve one or more replicate files on the database.

To control overall ETP operation an *administration file* contains the definition of all master/replicate file definitions, their relationships and physical locations. The administration file, which should be located on every nucleus having a master file, is the source of all information used by the ETP Replication Tasks to perform the actual master/replicate file resynchronization.

The log, confirmation and administration files can all be on one Adabas file when all master and replicate files are on the same physical database. However, the files are presented in this document as separate files to depict a more flexible ETP environment.

Replication Task Control

The ETP Replication Tasks actually resynchronize the replicate files with their master files. Each time a replication task is started or restarted, it checks the log files for the master files in the administration file. If the log files contain uncompleted changes-that is, logged changes not yet applied to the replicate files, the task applies those changes to the corresponding replicate files.

The first resynchronization occurs when a task is first started. A task can be started in one of three ways:

- Manually by the ETP administrator;
- By a user program invoked after each ET on the master file;
- At regular time intervals.

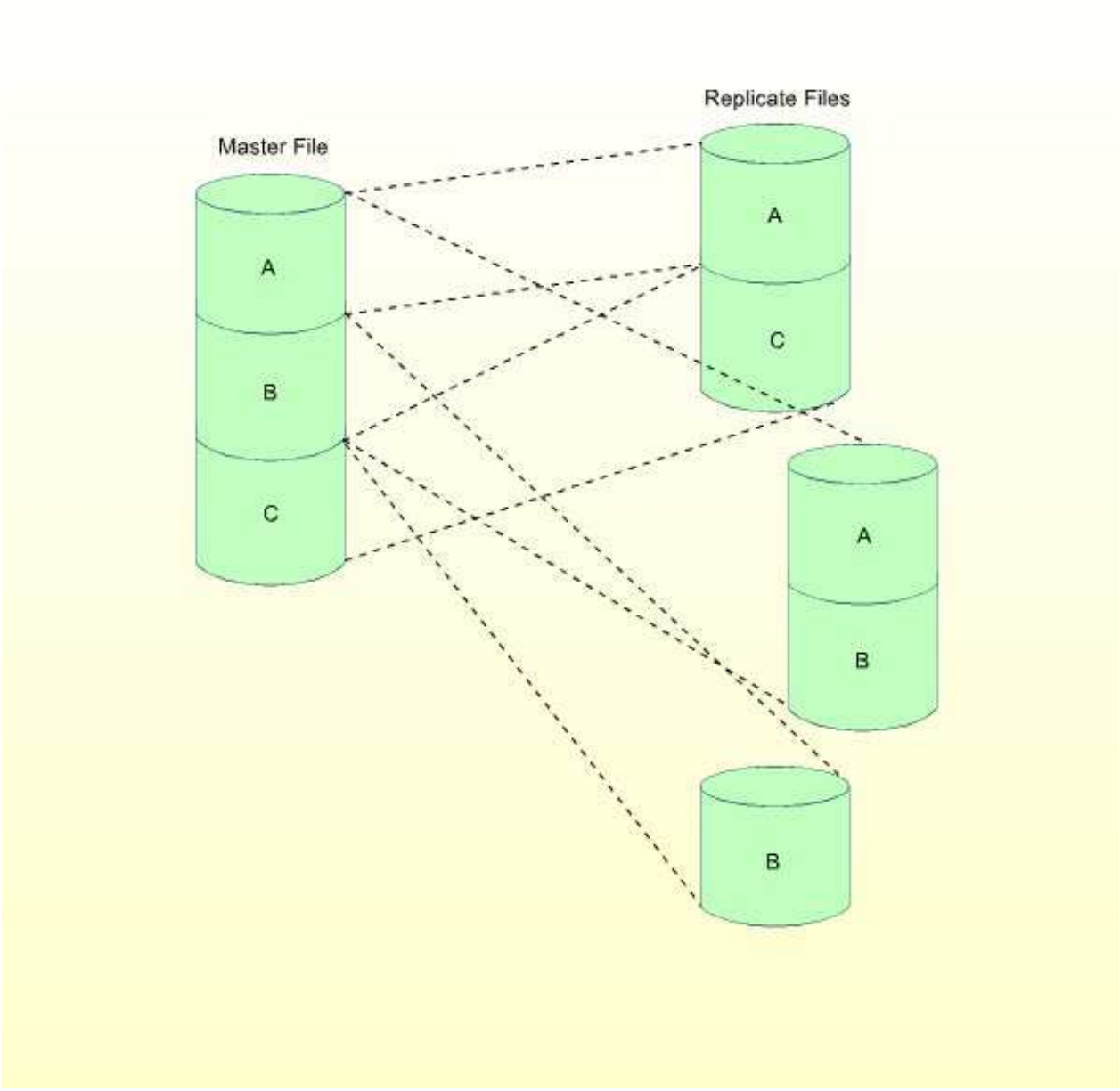
When or how often a task restarts depends on the values set at installation or by the ETP administrator. Tasks are stopped or restarted and/or placed under ET program or self-restart interval control using the ETP maintenance utility.

The maintenance utility, a menu-driven facility for setting up and managing ETP operation, allows you to:

- Define master and replicate files;
- Control Replication Task operation;
- Set up user/system profiles;

For further details, refer to the *ETP maintenance utility documentation*.

Replicate Files from Master File Record Subsets



Tailoring and Tuning ETP

Although the possibility of inconsistency between the master and the replicate files is theoretically greater than with the two-phase commit concept, ETP offers "tailoring and tuning"of the individual replicate copies, which can remove virtually any chance for differences to occur.

The "tailoring"ability of ETP allows you to define each replicate file copy for the needs of a particular location. In other words, the replicate file can contain either all data in the master file, or only the data needed at a particular location, as shown in the figure Replicate Files from Master File Record Subsets. This can reduce the need for "refreshing" the replicate file, since only a portion of the complete master file is represented.

The "tuning" ability in ETP allows you to control exactly when master/replicate synchronization occurs. You determine when resynchronization occurs by specifying when to start the ETP replication tasks. The choices for task start are:

- Manually at any time using the task control facilities of the maintenance utility;
- After each successful end-of-transaction (ET) on a master file, using your own program. You can make the ET control effective for specific master files (and their related replicate files);
- At regular self-restart intervals defined for all log files and the replicate databases to which the related master files are replicated. Multiple tasks can be run in parallel to:
  - Process different sets of log files;
  - Replicate to different replicate databases;
  - Have different self-restart intervals.

Resynchronization Modes

The following examples show how ETP operates in the three resynchronization modes:

- Manual Control
- ET Program Control
- Self-Restart Interval Control

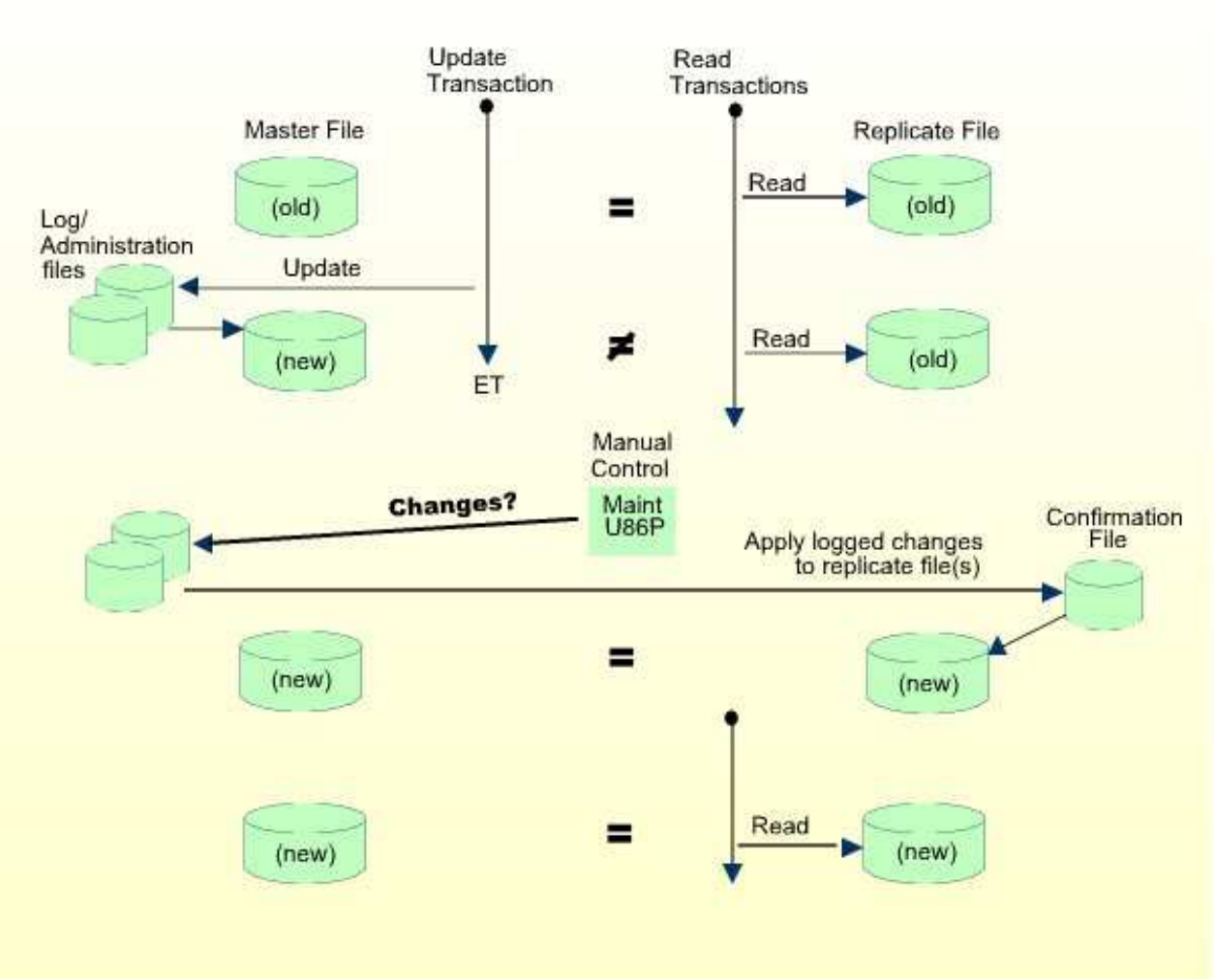
Manual Control

The figure below shows the basic operating sequence for a simple ETP environment of one master and one replicate file. The first two read transactions access the "old" replicate file. The single update transaction updates the "old" master file to "new" status.

After the change is actually applied to the master file, the program issues an end-of-transaction (ET) to denote that the update is complete.

By default, a task is executed only once to resynchronize master and replicate files. If the task is executed with a self-restart interval, all master and replicate files defined in the related administration file are resynchronized. Manual control is intended for resynchronizing of the ETP environment online.

Master/Replicate File Synchronization (Manual Control)

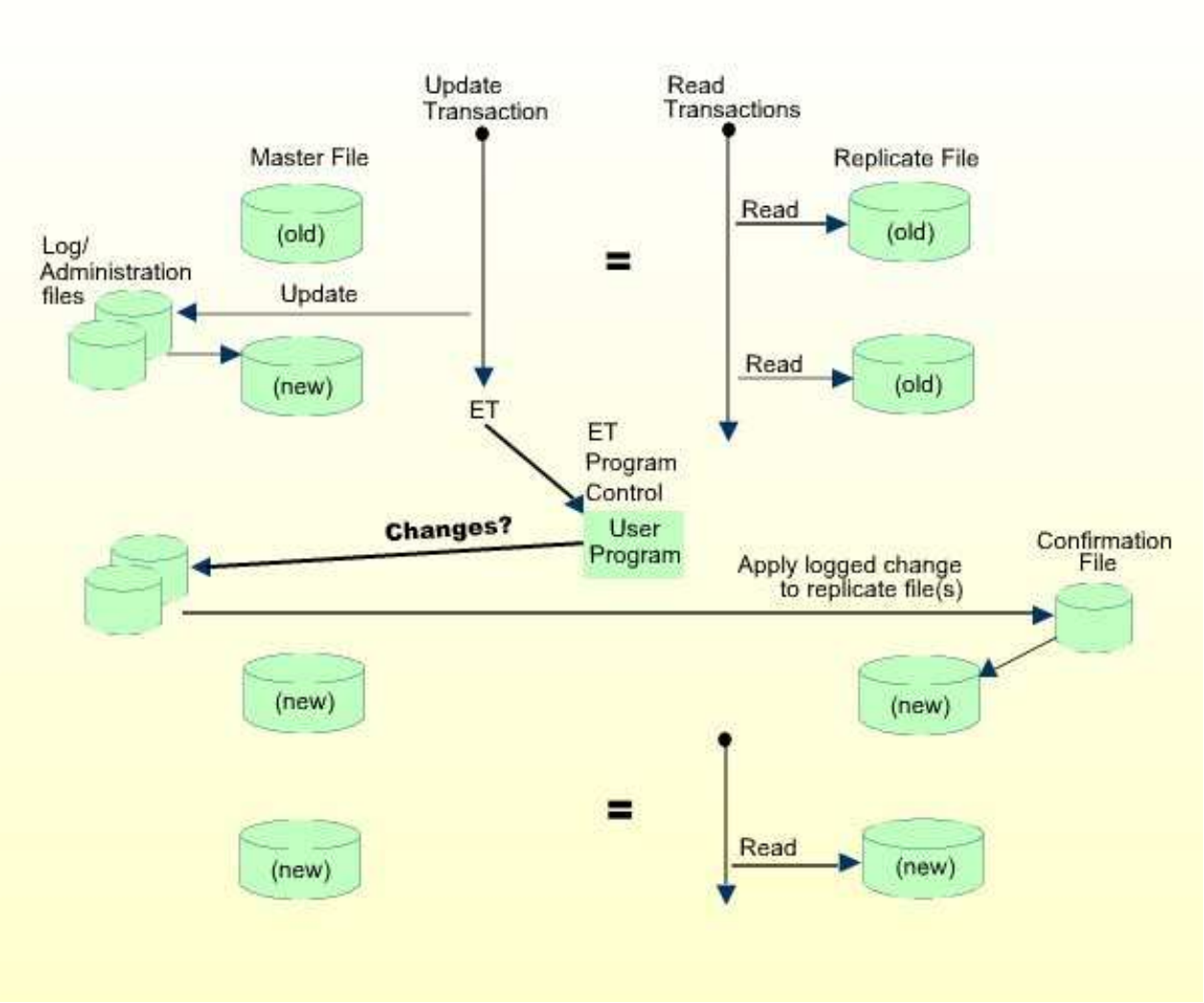


If all tasks have been stopped by means of the "Control Tasks" direct command, task execution for the current administration file is disabled; that is, no tasks can be executed. After task execution has been re-enabled, the tasks must be started manually by the ETP administrator.

ET Program Control

The figure below shows the ETP operation when the Adabas End-of-Transaction (ET) controls task operation.

Master/Replicate File Synchronization (ET Program Control)



The program must first be defined to and enabled by ETP as described under *Programming ETP*. The parameters passed to the program are described under *ET-Time User Program*.

Each master file that runs under ET program control must enable the program control. If a master file neither invokes ET program control nor operates under self-restart interval control, it and its replicate files can only be resynchronized under manual control.

If any of the master files which are updated in a transaction have enabled ET program control, the user-defined program is started after the ET command has been successfully executed.

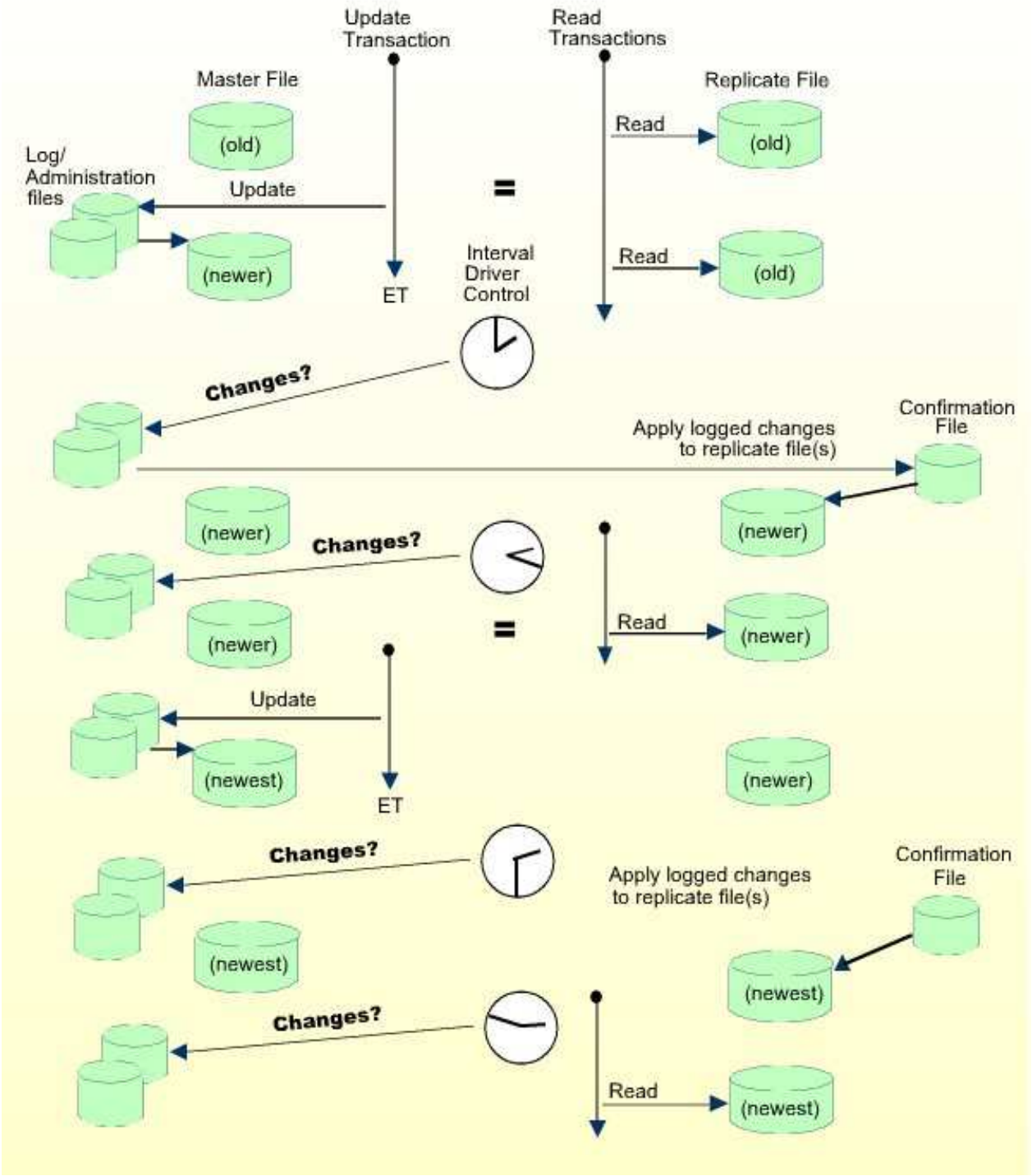
Self-Restart Interval Control

A fully automatic method of operation is the self-restart interval control. By defining and enabling an "elapsed time" between restarts, you can force Replication Tasks to resynchronize master files with replicate files. The figure below shows an example of this type of operation.

To run ETP in self-restart mode, you must define a time interval and then enable self-restart operation. You can define a different self-restart interval for every replication task.

Warning:
Self-restart resynchronization is intended for batch operation, and is not recommended for online operation; the terminal is locked during self-restart resynchronization. Self-restart resynchronization must not be used when running UTM, TIAM or IMS because a processor loop may result.

Master/Replicate File Synchronization (Interval Control)


Update Transaction
Read Transactions
Master File
Replicate File
(old)
Read
(old)
Log/ Administration files
Update
=
(newer)
ET
Interval Driver Control
Read
(old)
Changes?
Confirmation File
Apply logged changes to replicate file(s)
(newer)
(newer)
Changes?
(newer)
=
Read
(newer)
Update
(newer)
(newest)
ET
Changes?
Confirmation File
Apply logged changes to replicate file(s)
(newest)
(newest)
Changes?
Read
(newest)