Processing and Performance

The Adabas trigger driver is executed as a part of the Adabas nucleus. It generally controls the whole run-time processing of a trigger. It determines whether a trigger is to be fired, initiates the Natural trigger driver, and interacts with it to ensure the correct and timely processing of the procedures.

This chapter covers the following topics:

- Initialization
- Checking for Procedures
- Processing the Procedures
- Processing the Results
- Shutdown
- Abnormal Termination
- Command Logging

Initialization

When the Adabas nucleus starts, it determines whether the ADARUN parameter SPT=YES has been specified; if so, it passes control to the Adabas trigger driver to allow it to initialize. During initialization, the Adabas trigger driver performs the activities described in the following paragraphs.

Verifying the Adabas Triggers Profile

The Adabas trigger driver verifies the Adabas triggers profile on the trigger file and extracts the session parameters to be used in processing triggers and stored procedures for the session. If no profile exists on the trigger file, the initialization of the Adabas triggers and stored procedures facility in the nucleus is terminated with an appropriate error message.

- The Adabas trigger driver checks the "triggers status" and "stored proc. status" parameter settings in the profile.

Triggers	Stored Procedure	Action
Inactive	Inactive	The Adabas triggers and stored procedures facility is not allowed to start. To the Adabas trigger driver, the "inactive" setting has the same meaning as the ADARUN parameter setting SPT=NO.
Active	Inactive	Triggers processing is started but any request to run a stored procedure is rejected with a response code 22.
Inactive	Active	Stored procedures processing is started but any request to run a trigger is rejected with a response code 22.

- The Adabas trigger driver obtains information about the Natural subsystems from the profile:
- The subsystem name, which must correspond to the name of the linked Natural nucleus that will process the procedures; that is, the name given to the Natural nucleus during Part III of the installation procedure.
- The number of subsystems that must be started in order to handle the work load generated by the triggers that will be fired.

Verifying the Presence of at Least One Trigger

The Adabas trigger driver verifies that the trigger file contains at least one trigger definition. If not, the Adabas triggers and stored procedures facility is not initialized and an appropriate error message is given, even if triggers status is set to "inactive".

Acquiring Storage

The total storage requirement for the Adabas triggers and stored procedures facility depends on

- the size of the work areas required;
- the buffer sizes required for the trigger table, pre- and post-trigger queues; and
- the space needed for the Natural subsystems.

The amount of space needed for the Natural subsystems is determined by the size of the Natural nucleus and the various buffers that the Natural environment needs; for example, ESIZE, DATSIZE, and FSIZE.

If the size of the overall region/address space for the Adabas nucleus is too small, the Natural subsystems will not be able to run. The error message will normally indicate "insufficient storage" or an abend of the subsystems (response code 40000109 or 40000008 may be returned). If this occurs, increase the size of the region/address space or decrease the number of Natural subsystems used to execute the procedures.

Software AG recommends splitting the Natural nucleus to minimize nucleus storage requirements.

Creating the Trigger Table

At least one trigger must be defined on the trigger file; otherwise, processing cannot continue.

After the Adabas trigger driver determines the validity of the trigger file, trigger definitions are read and entries are added to the trigger table. In a nucleus cluster environment, the trigger table is not reread for each nucleus, but is obtained from a nucleus that is already active.

In order to maintain the integrity of the system, the trigger table is not updated with new, modified, or deleted triggers unless a REFRESH command is issued from the Trigger Maintenance facility (see the section *Updating the Trigger Table*).

The trigger table enhances performance. Instead of checking the trigger file itself for the existence of a trigger every time a command is processed, the Adabas trigger driver simply checks the trigger table, i.e., the buffer in memory. The sequence of the table enables the trigger driver to rapidly determine whether a trigger should be fired.

When reading the trigger file to determine entries for the trigger table, the Adabas trigger driver

- ignores any entries for the trigger, checkpoint, or security files;
- loads deactivated triggers, but ignores them until they have been activated from the Modify Trigger function in the Trigger Maintenance facility. See the section *Single Trigger Definition*.
- determines the maximum file number for the database (the highest file number used plus 10), and ignores any trigger for a file number greater than the maximum.

Ignoring out-of-range file numbers may cause a problem when the REFRESH command is used. One solution is to load a dummy file with a file number greater than the real maximum file number. You can then add new files that have a file number greater than the real maximum but less than the dummy file number. By having a fixed-size buffer established at nucleus initialization, storage requirements for this buffer are minimized. With a two-byte file number, the total maximum size could be very large.

Starting Natural Subsystems

After the Adabas trigger driver is initialized, it starts the Natural subsystems that are responsible for the actual execution of the procedures. The "maximum subsystems" parameter in the Adabas triggers profile determines the number of subsystems (1-10) to be started.

Each subsystem is typically a minimally modified batch Natural nucleus that runs in the Adabas address space. This affects the region size specified on the MPM start-up JCL/JCS.

When a subsystem is started, the Adabas trigger driver keeps track of any change in subsystem status or activity; hence, each subsystem can uniquely identify itself to either the Adabas trigger driver or any procedure that the subsystem invokes. The user can monitor these activities by using the Subsystem Activity function, which is part of the Trigger Maintenance facility.

When a Natural subsystem becomes active:

- the Natural trigger driver gets control; and
- the Adabas trigger driver is informed that the subsystem is ready to start processing any procedures that may result from a stored procedure request or the firing of a trigger.

The subsystem queue contains an entry for each Natural subsystem that is waiting for work. When the Adabas trigger driver needs a subsystem, it examines the subsystem queue to find one that is available.

Checking for Procedures

Once the Adabas nucleus is initialized, user processing continues normally. For each command that the nucleus receives, the Adabas trigger driver determines whether a trigger needs to be fired. (Entries in the trigger table that are marked "inactive" are ignored.)

For pre-command triggers, the Adabas trigger driver checks for triggers before the command is selected for processing by the Adabas thread. This involves the READ, FIND, STORE, DELETE, and UPDATE commands. For these commands, the Adabas trigger driver determines whether there are any triggers to be fired; if not, command processing continues normally. Commands like end transaction (ET), close (CL), release command ID (RC) are not checked but are given directly to the nucleus for normal processing. The trigger check is, of course, not done for stored procedure requests.

Once a command has been processed successfully by Adabas and the response code is zero, the Adabas trigger driver determines whether there are any post-command triggers to be fired. If not, the user is informed in the usual manner. If pre- or post-command trigger checking *does* result in a trigger being fired, the Adabas trigger driver can proceed with the trigger processing.

Scanning the Trigger Table

Once it has been determined that a command is eligible for the firing of a trigger, the trigger table is scanned. For performance reasons, the order in which triggers are scanned is determined by the sequence or priority assigned to the triggers by the user (see the section *Multiple Trigger Definitions*).

If two triggers exist for the same command class and have the same priority, they are scanned in the order in which they are read from the trigger file (that is, the ISN sequence of the records on the file). It is therefore important to specify the priority for each trigger correctly.

Triggers are scanned in the following general sequence:

Sequence	Field	Command
1	specific field	specific command
2	any field	specific command
3	specific field	any command
4	any field	any command

Creating Pre- and Post-Command Trigger Queue Entries

If a command results in a trigger being fired, or if the Adabas trigger driver determines that the command is a stored procedure request, an entry is created in the

- pre-command trigger queue if the command has not been executed; or the
- post-command trigger queue if the command has been executed successfully.

The entry contains information obtained from both the command that caused the trigger to be fired and the corresponding entry in the trigger table (for example, details about the procedure to be executed). The entry also allows the Adabas trigger driver to keep track of the status of triggers that are fired.

If a Natural subsystem is waiting for work, it is given the trigger request immediately. Otherwise, the trigger request remains in the pre- or post-command trigger queue until the next subsystem is available.

Processing the Procedures

When a trigger request is placed in the pre- or post-trigger queue and a subsystem accepts it, processing continues under the control of the Natural trigger driver.

Only two triggers (one pre- and one post-command trigger) can be fired for any one command, regardless of the results.

When a command results in a trigger being fired, the system checks whether the trigger is asynchronous or synchronous.

Asynchronous Triggers

If the trigger is asynchronous, the command does not wait for the triggered procedure to complete. The command is released and processing continues normally depending on whether the trigger is pre- or post-command:

pre-command	The command is made available for processing in the Adabas thread. The triggered procedure may be processed after the command has executed or simultaneously with command execution.
post-command	The triggered procedure and the command are processed independently. The user is informed when the trigger is fired, regardless of the results of the procedure.

Synchronous Triggers

If the trigger is synchronous (participating or non-participating), the command is held until the Natural trigger driver notifies the Adabas trigger driver that the execution of the procedure is complete.

If the return code is zero, the command is released to continue processing.

If the return code is non-zero, with user receives a response code 155 or 156 and the following information:

- the additions 3 field contains the name of the procedure that was executed as a result of the trigger being fired.
- the first two bytes of the additions 4 field contain the actual return code from the procedure.
- the second two bytes of the additions 4 field contain a subcode that indicates the type of trigger that was fired:
  - subcode 15 indicates a pre-command trigger.
  - subcode 16 indicates a post-command trigger.

Processing the Results

After the procedure executes, the results are placed in the trigger request entry and the status is updated appropriately. When the Adabas trigger driver detects this, it "finalizes" the trigger processing of the command.

For both pre- and post-command triggers, the return code from the procedure determines how the results are processed as described in the following sections.

Pre-Command Triggers

Return Code	Action
zero	the command is "released" so that it can be executed.
non-zero	the command is not executed and the user receives response code 155 in the response code field of the Adabas control block.

Once the command has been executed by the Adabas thread, it may be reselected for any post-command trigger processing.

Special Processing for Synchronous Pre-Command Triggers

For synchronous triggers, a return code of "1" indicates that the procedure completed processing successfully. The response code field in the Adabas control block is set to zero to indicate the successful result. However, the command is not "released" for execution by the nucleus; instead, the results of the procedure are immediately returned to the user.

This special processing accommodates procedures that have read/write access to the record buffer and require the command to be processed in a way that is similar to a stored procedure. See the section *Using the Format and Record Buffers*.

It is possible for a procedure of a pre-command trigger to modify the contents of the record buffer before the command is executed; this can be useful with update and store commands.

Post-Command Triggers

Return Code	Action
zero	the command is considered to be successful and the user is informed with response code 0 (zero) in the Adabas control block.
non-zero	response code 156 is returned in the Adabas control block. Although the procedure returned a non-zero code, the actual command may have been successful; the results of the command execution must be interpreted by the application that issued the command.

Note:
When a post-command trigger is fired and the return code from the procedure is non-zero, the data in the record buffer is not returned, even if the command was executed successfully.

Special Processing for Synchronous Post-Command Triggers

Whether it is successful or not, a post-command trigger that is synchronous and has read/write access to the record buffer may have modified the record buffer.

In the case of participating triggers, the results of the trigger may have changed the result of the command. For example, if a successfully executed UPDATE command fires a post-command trigger and the procedure for the trigger does not complete successfully, it may or may not perform a BT command. When the user is informed, response code 156 is given. The application that issued the original command must determine whether the UPDATE command is still in effect, and perform the appropriate action (ET or BT).

Shutdown

Shutdown may occur in the following situations:

- The Adabas trigger driver keeps track of the subsystems that fail. If *all* subsystems fail, it determines that no further processing of procedures is possible and terminates. Shut-down processing depends on the "error action" value (see the table *Error Action*).
- The nucleus receives the ADAEND or HALT operator command and instructs the Adabas trigger driver to shut down as well:

HALT	the Adabas trigger driver terminates immediately.
ADAEND	the Adabas trigger driver terminates when all subtask activities are at ET status. If a subtask is busy and is involved in ET logic, it is allowed to finish if it is issuing commands to the current nucleus; otherwise, a message is displayed at the console (see message ADAN9M) and the subtask is terminated with the transaction incomplete.

Shut-down Processing Steps

Shut-down processing steps are as follows:

1. The pre- and post-trigger queues are checked for any waiting triggers. Response code 148 is issued to all users who are waiting for a synchronous trigger to complete execution.
2. The user is informed in the normal manner for any completed post-command triggers.
3. Response code 148 is issued for completed pre-command triggers because they will not be processed by the Adabas thread. If these commands are part of an ET transaction, the user should issue a BT and ET command as appropriate.
4. Response code 157 ("command is rejected") is issued for any post-command trigger that is detected after the shutdown begins. The command was executed before shutdown began, but the triggered procedure will not be executed.
5. Any subsystem that remains active after five seconds is forced to terminate, and a message is displayed at the console. A subsystem is considered "active" if it contains a procedure that continues to run; the Natural programs are in the buffer pool and the program may be issuing database calls to another database or no database at all. In this situation, a "halt" issued to the current database may be ineffective.
6. All subsystems shut down.
7. The total numbers of triggers and stored procedures are written to the console, and the triggers status field in the Adabas triggers profile is set to "inactive". The nucleus continues shut-down processing in the normal way.

Error Action

If the shutdown is requested by the Adabas trigger driver itself, shut-down processing depends on the value assigned to the error action field in the Adabas triggers profile:

Action	Shut-down Processing
Halt	The nucleus must also be terminated. The ADAEND request originates from the Adabas trigger driver itself. Any user application still processing in the subsystem is terminated.
Ignore	The Adabas triggers and stored procedures facility terminates in the Adabas nucleus, but nucleus processing continues in the normal manner (as if the ADARUN parameter SPT=NO were specified). No triggers will be fired to perform any extended command processing, and certain integrity problems may result.
Reject	Any command that would normally result in a trigger being fired receives response code 157. The Adabas triggers and stored procedures facility remains active; however, all subsystems are shut down and procedure processing is discontinued.

In a nucleus cluster environment, when one nucleus is set to Ignore or Reject status, all nuclei in the cluster are also set to this status.

Abnormal Termination

See the section *Shutdown* for information about situations that shut down

- the Adabas nucleus, the Adabas trigger driver, and all subsystems;
- the Adabas trigger driver and all subsystems, but not the Adabas nucleus;
- all subsystems, but not the Adabas triggers and stored procedures facility or the Adabas nucleus.

Natural ESTAE / STXIT Processing

If DU=OFF (the default; no memory dump is generated for an abend) is specified in the NATPARMs, the Natural ESTAE / STXIT is active for the duration of the session.

If a program abend occurs within the Natural subsystem when the Natural ESTAE / STXIT is active, the abend is trapped: the Natural ESTAE / STXIT exit acquires control, cleans up, notifies the Adabas trigger driver, and restarts the Natural trigger driver.

Thus, the Natural session is restarted instead of terminated. This is an important performance consideration.

If DU=ON is used, the ESTAE / STXIT is not activated and the subsystem will terminate abnormally. Performance is slowed if the Adabas trigger driver must restart because it will terminate and restart the subsystem as appropriate.

Natural Subsystem Abends

An executing procedure may exceed the time-out limit, or be cancelled by the DBA before it completes processing. (Time-out refers to elapsed time as opposed to CPU time usage.)

These two "abnormal" terminations of a procedure are similar, with the following exceptions:

- A cancellation is done manually from the Trigger Maintenance facility.
- A time-out occurs automatically based on the activity timeout setting in the Adabas triggers profile.

An executing procedure may need to be terminated at any stage, i.e., waiting, looping, or simply executing longer than is expected. The only sure way to intercept processing is to terminate the subsystem itself. (You can observe this termination by monitoring the subsystem from the Trigger Maintenance facility; see the section *Subsystem Activity*.)

If the trigger is synchronous, the user must be informed that the subsystem has been terminated. Depending on the trigger type (pre- or post-command), response code 155 or 156 is set, with subcode 9 to indicate the timeout. The user is always informed, whether the subsystem timed out or was deliberately terminated, and additional information is provided in messages that are written to the console.

When an abnormal termination occurs, ascertain the reason and correct the problem. If this happens continuously, deactivate the trigger until the problem has been solved. Use the trigger activity logging option on the profile to determine the reason more easily.

Natural Subsystem Restart

If Natural is unable to recover and a subsystem terminates, the Adabas trigger driver is notified of the error and a message is written to the console.

After a Natural subsystem is terminated, it is restarted automatically.

- If the restart is successful, the subsystem is reinitialized in the usual way. If there is not currently work to be done, the subsystem is placed in a queue to wait until the Adabas trigger driver informs it that a new trigger was fired and the procedure needs to be processed.
- If the restart is unsuccessful, it is attempted two more times. The subsystem is permanently deactivated if it fails to start after three consecutive restart attempts, and a message is displayed at the console to inform the user. The subsystem cannot be reactivated until the nucleus is shut down and restarted.

The restart routine responsible for restarting the Natural trigger driver is STP.

Users who are waiting for a synchronous trigger to finish processing are notified that the trigger did not complete successfully. Response code 155 or 156 is returned, with additional information in the additions 4 field.

Command Logging

Because a triggered procedure may reject a command, it is possible when running Adabas triggers and stored procedures that a command issued by the user is never run in an Adabas thread or even seen by the nucleus supervisor.

It is therefore necessary for the Adabas triggers and stored procedures facility to log

- a PC command (that is, a stored procedure request) when it is received;
- a command when it is selected for a pre-command or post-command trigger;
- a pre- or post-triggered command when it receives a non-zero response.

When processing a command log record, the log record types are

- X’0005’ for pre-command triggers; and
- X’0006’ for post-command triggers.

For the Adabas control block in the log record, the additions 3 field contains the name of the procedure being invoked and the additions 4 field contains the following information about the procedure:

Byte	Contains...
1-2	the response code from the procedure.
3	"P" for pre-trigger; "R" for procedure call; or "S" for post-trigger.
4	"A" for asynchronous; "N" for non-participating; or "P" for participating.
5	"R" for read; "F" for find; "U" for update; "S" for store/add; "D" for delete; or "P" for procedure call.
6	flag settings: "1" for no parameters; "2" for response code only; "4" for control information; "8" for special stored procedure parameters; "10" for record buffer access; or "80" for record buffer update.
7-8	the name of the field associated with the trigger.

Note:
No log is created for an asynchronous trigger that returns a non-zero response. The CQE address (4th parameter) for an asynchronous trigger is set to zero if the command completes before the trigger is processed.