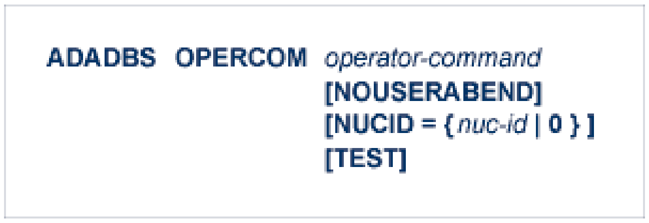
# OPERCOM: Issue Adabas Operator Commands

The OPERCOM function issues operator commands to the Adabas nucleus.

In an Adabas cluster environment, OPERCOM commands can be directed to a single cluster nucleus or to all active nuclei in the cluster. If a particular nucleus is not specified, the command defaults to the local nucleus.

Adabas issues a message to the operator, confirming command execution.



```
ADADBS  OPERCOM  operator-command
                 [NOUSERABEND]
                 [NUCID = { nuc-id | 0 } ]
                 [TEST]
```

In this section, the discussion of the individual operator commands follows the discussion of the optional parameters, since some of the operator commands behave differently when issued in an Adabas cluster environment.

This chapter covers the following topics:

---

## Using OPERCOM Commands in Cluster Environments

Some ADARUN parameters are *global parameters*; that is, they must have the same values for all nuclei in a cluster. Of these, some are set at session initialization and cannot be changed. Others can be modified on a running system. OPERCOM commands that change these modifiable global parameter values are handled in a special way in cluster environments.

If an Adabas cluster nucleus changes one or more global parameters, that nucleus acquires a *parameter change lock*, makes the changes in its local parameter area, informs the other cluster nuclei of the changes and waits for a reply. The other cluster nuclei make the changes in their own local parameter areas and send an acknowledgment message.

## Optional Parameters

**GLOBAL: Operate Across All Active Cluster Nuclei**

Five OPERCOM commands use the GLOBAL option to operate across all active nuclei in a cluster: ADAEND, CANCEL, FEOFCL, FEOFPL, and HALT. For example:

```
ADADBS OPERCOM ADAEND, GLOBAL
```

All other OPERCOM commands use the NUCID=0 option to operate across all active nuclei in a cluster.

### NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

**Note:**
When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

### NUCID: Cluster Nucleus ID

Any nucleus running in an Adabas nucleus cluster is allowed to run Adabas utilities such as ADADBS.

With certain exceptions, the NUCID parameter allows you to direct the ADADBS OPERCOM commands to a particular nucleus in the cluster for execution, just as though the command had been issued by a locally run ADADBS OPERCOM operation. You can route most OPERCOM commands to all nuclei in a cluster by specifying NUCID=0.

If NUCID is not specified in a cluster environment, the command is routed to the local nucleus.

**Note:**
For ADADBS OPERCOM and Adabas Online System (AOS), a zero value for the NUCID parameter indicates that the command applies to all nuclei in the cluster (global). A nonzero value for the NUCID parameter indicates that the command applies only to the cluster nucleus specified.

### TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; nor the validity of values and variables. See Syntax Checking with the TEST Parameter for more information on using the TEST parameter in ADADBS functions.

## Operator Commands

The following Adabas operator commands can be used in the ADADBS OPERCOM utility function:

-
  ```
  ADAEND [, GLOBAL ]
  ```

  This command terminates an Adabas session normally. No new users are accepted after this command has been issued. ET logic updating is continued until the end of the current logical transaction for each user. After all activity has been completed as described above, the Adabas session is terminated.

  In nucleus cluster environments, the GLOBAL option can be used to terminate the Adabas session in all active cluster nuclei.

-
  ```
  ALOCKF = file-number
  ```

  Lock a file in advance to ensure that an EXU, EXF, or UTI user will have exclusive control of the specified file. The advance-lock prevents new transactions from using the file. Once all current users have stopped using the file, the exclusive-control user has the lock. Until then, the exclusive-control user must wait.

  To remove the advance lock without running the utility, see the RALOCKF command.

  This command is not available in single user mode or for a read-only nucleus. It is available in cluster and non-cluster environments.

  The following key points should be noted about advance-locks on files:

  1. An advance-lock can be set while a file is being used.
  2. A command requesting exclusive control (UTI, EXF, or EXU) over an advance-locked file will wait in the command queue until all other users stop using the file before it starts processing the file.
  3. Advance-locks are automatically removed when a user gets exclusive control over the file. However, if a file is locked (via the LOCKF, LOCKU, or LOCKX commands), the locks are not removed when a user gets exclusive control over the file. (Locks must be explicitly removed, whereas advance-locks are automatically removed.)
  4. Adabas will reject an advance-lock on a file that is already locked (via the LOCKF, LOCKU, LOCKX or ALOCKF commands) but will accept a lock request on an advance-locked file.
  5. To ensure you have uninterrupted exclusive control over a file in a situation where you have multiple steps to run that require uninterrupted exclusive control while all steps have been processed, use a combination of advance-locking the file (ALOCKF), stopping all users of the file (STOPF), and locking the file (LOCKU). An example of this is given later in this section.

6. In the case of expanded files, an ALOCKF command is applied to the anchor file (representing the entire expanded file chain).
7. In a cluster environment, advance-locks are effective in all nuclei of the cluster.

### Simple Example

In the following example, issuing the ALOCKF request to advance-lock file 32 ensures that file 32 will be available so the ADALOD UPDATE function can take exclusive control (via a UTI request) of the file for its processing:

```
ADADBS OPERCOM ALOCKF=32
ADALOD UPDATE FILE=32
```

Adabas processing proceeds in the following manner for these utility functions:

1. When the ADADBS OPERCOM ALOCKF request is submitted, file 32 is marked as advance-locked.
2. If there are any active users of file 32, the ADALOD UTI request cannot be granted immediately and will wait for the active users to end their transactions or sessions. Active users continue to issue commands against file 32. However, requests by new users for file 32 are rejected because of the advance-lock on the file.
3. When all active users of file 32 have ended their transactions or sessions, the ADALOD UTI request for exclusive control can be granted. Once exclusive control is established, ADALOD UPDATE processing can occur.

   As part of the successful execution of the ADALOD UTI request, the advance-lock is removed from the file. However, because ADALOD processing now has exclusive control of file 32, other users still cannot access it.

To accelerate the process and limit the wait time for the ADALOD UTI request, you can simply stop all active users of the file by force using the STOPF operator command:

```
ADADBS OPERCOM ALOCKF=32
ADADBS OPERCOM STOPF=32
ADALOD UPDATE FILE=32
```

In this case, the STOPF command will cause the nucleus to back out and stop users of file 32 before the ADALOD UTI request is granted. In addition, the advance-lock request specified by the ALOCKF command will prevent new users from accessing the file until the ADALOD UTI request for exclusive control is processed.

### More Complex Example

An advance-lock set by ALOCKF cannot guarantee that multiple job steps in a series get uninterrupted exclusive control over a file, as the advance-lock is removed when the first step obtains exclusive control. Suppose an installation wants to run the following utility sequence:

```
ADAULD UNLOAD FILE=45
ADADBS REFRESH FILE=45
ADALOD UPDATE FILE=45
```

An ALOCKF request to advance-lock file 45 in this case would only work for the ADAULD UNLOAD function, because the ADAULD EXU request for exclusive control of file 45 would remove the advance-lock. If there are active users who try to issue commands against file 45, there is a chance that one of them will execute a command between the UNLOAD and REFRESH steps, or between the REFRESH and UPDATE steps. Such a user may also prevent the REFRESH or UPDATE step from obtaining exclusive control of file 45.

To ensure you have uninterrupted exclusive control over the file in this situation, use a combination of advance-locking the file (ALOCKF), stopping all users of the file (STOPF), and locking the file (LOCKU):

```
ADADBS OPERCOM ALOCKF=45
ADADBS OPERCOM STOPF=45
ADADBS OPERCOM LOCKU=45
ADAULD UNLOAD FILE=45
ADADBS REFRESH FILE=45
ADALOD UPDATE FILE=45
ADADBS OPERCOM UNLOCKU=45
```

In this example, Adabas processing proceeds in the following manner:

1. When the ADADBS OPERCOM ALOCKF request is submitted, file 45 is marked as advance-locked.

   The ADADBS OPERCOM STOPF request causes the nucleus to back out and stop users of file 45. (This step is optional.)

   The ADADBS OPERCOM LOCKU request locks the file more permanently than the ALOCKF request since the LOCKU lock will stay in effect until it is explicitly released.

2. If there are any active users updating file 45, the ADAULD EXU request cannot be granted immediately and will wait for the update users to end their transactions or sessions. Active users may continue to issue commands against file 45. However, requests by new users for file 45 are rejected because of the advance-lock on the file.

3. When all active users of file 45 have ended their transactions or sessions, the ADAULD request for exclusive-update (EXU) control can be granted. Once exclusive-update control is established, ADAULD UNLOAD processing can occur.

   As part of the successful execution of the ADAULD EXU request, the advance-lock is removed from the file. When ADAULD completes processing, it releases the EXU control of file 45. However, during and after the ADAULD execution, the LOCKU lock ensures that other users still cannot access the file.

4. The ADADBS utility will issue a UTI request for exclusive control of file 45, which will be granted. ADADBS REFRESH processing will then occur. When it completes, ADADBS will release exclusive control of file 45. However, the LOCKU lock ensures that other users still cannot access it.
5. The ADALOD utility will issue a UTI request for exclusive control of file 45, which will be granted. ADALOD UPDATE processing will then occur. When it completes, ADALOD will release exclusive control of file 45. However, the LOCKU lock ensures that other users still cannot access it.
6. The ADADBS OPERCOM UNLOCKU request explicitly unlocks file 45, making it available for other users.

-

  ```
  CANCEL [, GLOBAL ]
  ```

  Cancel the Adabas session immediately. All command processing is immediately suspended. A pending autorestart is in effect which in turn causes the autorestart routine to be executed during the initialization of the next Adabas session.

  In nucleus cluster environments, the GLOBAL option can be used to cancel the Adabas session in all active cluster nuclei.

-

  ```
  CLOGMRG = { YES | NO }
  ```

  Switches automatic command log merging (ADARUN CLOGMRG parameter value) on or off in nucleus cluster environments.

  The CLOGMRG command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

-

  ```
  CT = timeout-limit
  ```

  Dynamically override the ADARUN CT parameter value; that is, the maximum number of seconds that can elapse from the time an Adabas command has been completed until the results are returned to the user through interregion communication (which depends on the particular operating system being used). The minimum setting is 1; the maximum is 16777215.

  In nucleus cluster environments, the CT command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

-

**DAUQ**

Display the user queue element (UQE) of each user who has executed at least one Adabas command within the last 15 minutes.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

●

**DCQ**

Display all posted command queue elements (CQEs). Each CQE's user ID, job name, and buffer length is displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

●

**DDIB**

Display data integrity block (DIB). This block contains entries indicating which Adabas utilities are active and the resources being used by each utility. The DDIB function can be performed with either an active or an inactive nucleus.

In nucleus cluster environments, the information displayed by the DDIB command is global; the command can be run on any nucleus.

●

**DDSF**

Display Adabas Delta Save Facility Facility (DSF) status. The Adabas nucleus displays the DSF status on the operator console as well as in the ADADBS job protocol.

This function is only available if the nucleus is run with the parameter ADARUN DSF=YES.

In nucleus cluster environments, the information displayed by the DDSF command is global; the command can be run on any nucleus.

●

**DFILES=** { *n* | *n1* ,..., *n5* }

Displays the number of access, update, EXU, and UTI users for the specified files. User types are totaled for each file, and are listed by file. Up to five files can be specified in this command. Up to 798 users are displayed.

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DFILUSE =** *file-number*

  Displays the count of commands processed for the specified file so far during the current session.

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DHQA**

  Display up to 1000 hold queue elements.

- **DLOCKF**

  Display locked files.

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DNC**

  Display the number of posted command queue elements (CQEs) waiting to be selected.

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DNH**

  Display the number of ISNs currently in the hold queue.

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DNU**

  Display the number of current users.

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DONLSTAT**

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Display status of each active reorder or invert online process together with the process ID.

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DPARM**

  Display the Adabas session parameters currently in effect.

  **Note:**
  Additional Adabas add-on products and nucleus components will display additional parameters from those shown for a classic nucleus. For a sample of this report, please review the Adabas operator command documentation.

-

**DRES**

Display the allocated pool space and the highest use level ('high water mark') reached so far during the current session by record count and by percent for the following resources:

- attached buffers (AB)
- command queue (CQ)
- format pool (FP)
- hold queue (HQ)
- pool for the table of ISNs (TBI)
- pool for the table of sequential commands (TBQ or TBLES)
- user queue (UQ)
- unique descriptor pool (DUQPOOL)
- security pool
- user queue file list pool
- work pool (WP)
- pool for global transaction IDs (XIDs; nonzero only with Adabas Transaction Manager)
- cluster block update redo pool (nonzero only for a cluster nucleus with ADARUN LRDP greater than zero)
- Work part 1 area (WKP1)

  **Note:**
  The maximum pool value of Work part 1 is derived from the LP parameter. It corresponds to the maximum number of blocks a transaction can spend on Work Part 1 before Adabas decides to back it out.
- Work part 2 area (WKP2)
- Work part 3 area (WKP3)

The actual values are displayed in nucleus message ADAN28.

- **DSTAT**

Display the current Adabas nucleus operating status.

- **DTH**

  Display thread status.

- **DUQ**

  Display up to five active and inactive user queue elements.

- **DUQA**

  Display all user queue elements (UQEs).

  In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- **DUQE = X'*user-id*'**

  Display a user queue element for the specified Adabas-assigned user ID as follows:

  ```
  DUQE=X'A3CF2'
  ```

  The user ID must be entered in hexadecimal format. Do not use a job name for the user ID.

  In nucleus cluster environments, NUCID must always be specified because the user ID is not unique to the cluster.

- **DUUQE**

Display utility user queue elements (UQEs).

In nucleus cluster environments, the NUCID=0 option can be used to display information for all active cluster nuclei. Information is displayed for each nucleus, one after the other.

- 
  ```
  FEOFCL [, GLOBAL ]
  ```

  Close the current dual or multiple Command log and switch to the other dual or another multiple Command log. This command is valid only if dual or multiple command logging is in effect.

  In nucleus cluster environments, the GLOBAL option can be used to switch the dual or multiple command log in all cluster nuclei at the same time.

- 
  ```
  FEOFPL [, GLOBAL ]
  ```

  Close the current dual or multiple data Protection log and switch to the other dual or another multiple Protection log. This command is valid only if dual or multiple data protection logging is in effect.

  In nucleus cluster environments, the GLOBAL option can be used to switch the dual or multiple protection log in all cluster nuclei at the same time.

- 
  ```
  HALT [, GLOBAL ]
  ```

  Stop Adabas session. A BT (backout transaction) command is issued for each active ET logic user. The Adabas session is then terminated; no dumps are produced.

  In nucleus cluster environments, the GLOBAL option can be used to halt the Adabas session in all active cluster nuclei.

- 
  ```
  LOCKF = file-number
  ```

  Lock the specified file. The specified file will be locked at all security levels.

-

```
LOCKU = file-number
```

Lock the specified file for all non-utility use. Adabas utilities can use the file normally.

-
  ```
  LOCKX = file-number
  ```

  Lock the specified file for all users except EXU or EXF users. EXU and EXF users can use the file normally. The lock is released automatically when an EXU user issues an OP command.

-
  ```
  LOGGING
  ```

  Start command logging.

-
  ```
  LOGxx
  ```

  Begin logging as indicated by *xx* for each command logged where *xx* is one of the following:

  - CB - the Adabas control block
  - FB - the Adabas format buffer
  - IB - the Adabas ISN buffer
  - IO - Adabas I/O activity
  - RB - the Adabas record buffer
  - SB - the Adabas search buffer
  - UX - user data passed in the seventh parameter of the Adabas parameter list
  - VB - the Adabas value buffer
  - VOLIO - the extended I/O list for CLOGLAYOUT=5 and CLOGLAYOUT=8

-
  ```
  LOGWARN = { seconds | 0 }
  ```

Use the LOGWARN command to specify how often the PLOG and CLOG status is checked and resulting alert messages are produced. Valid values range from zero (0) through 2147483647 seconds. The default is 0, indicating that no PLOG or CLOG status checking occurs and no corresponding alert messages are produced. If a non-zero value is specified for LOGWARN, a valid user exit 2 or user exit 12 must also be specified.

- **NOLOGGING**

  Stop or prevent command logging.

- **NOLOG***xx*

  Stop or prevent logging of *xx* where *xx* is one of the following:

  - CB - the Adabas control block
  - FB - the Adabas format buffer
  - IB - the Adabas ISN buffer
  - IO - Adabas I/O activity
  - RB - the Adabas record buffer
  - SB - the Adabas search buffer
  - UX - user data passed in the seventh parameter of the Adabas parameter list
  - VB - the Adabas value buffer
  - VOLIO - the extended I/O list for CLOGLAYOUT=5 and CLOGLAYOUT=8

- **ONLRESUME = X'***identifier***'**

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Resume a previously suspended online reorder or invert process.

  In a cluster environment, NUCID must always be specified because the online process ID is not unique to the cluster.

- **ONLSTOP = X'***identifier***'**

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Stop an online reorder or invert process cleanly. The process continues up to its next interrupt point in order to produce a consistent state, and then terminates after performing all necessary cleanup.

  In a cluster environment, NUCID must always be specified because the online process ID is not unique to the cluster.

- **ONLSUSPEND = X'***identifier***'**

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Suspend an online reorder or invert process. The process continues up to its next interrupt point in order to produce a consistent state, performs a command throwback, and enters a state where it cannot be selected for processing. This command is useful if the online process is consuming too much of the nucleus resources.

  In a cluster environment, NUCID must always be specified because the online process ID is not unique to the cluster.

- **RALOCKF =** *n*

  Remove the advance lock on the specified file (see ALOCKF command) without running the utility.

  This command is available in cluster and non-cluster environments.

- **RALOCKFA**

  Remove the advance lock on all files for which it has been set (see ALOCKF command) without running the utility.

This command is available in cluster and non-cluster environments.

-
  ```
  RDUMPST
  ```

  Terminate online dump status. This command is normally used if online execution of the ADASAV utility has terminated abnormally.

-
  ```
  READONLY = { YES | NO }
  ```

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Switches READONLY status on or off.

  In nucleus cluster environments, the READONLY command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

-
  ```
  REVIEW = { NO | LOCAL | hub-id }
  ```

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Deactivate Adabas Review; change from hub mode to local mode; specify or change the Adabas Review hub with which a nucleus communicates.

-
  ```
  STOPF = file-number [, PURGE]
  ```

  Stop all users who are using the specified file. Any open transactions of the stopped users are backed out. Unless PURGE is also specified, a stopped user who returns (by sending a command) receives response code 9 (ADARSP009).

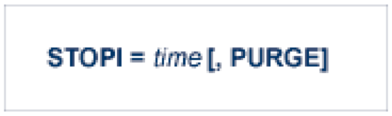
  If the optional PURGE parameter is specified, the stopped users are also deleted (their user queue elements are removed from the user queue).

  This command does not stop EXF or UTI users.

The following is an example of using the PURGE parameter:

```
ADADBS OPERCOM STOPF=5,PURGE
```

**Caution:**
If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPF command with PURGE only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur.

- 

  **STOPI =** *time* **[, PURGE]**

  Stop all users who have not executed a command during the specified time interval (in seconds). Any open transactions of the stopped users are backed out. Unless PURGE is also specified, a stopped user who returns (by sending a command) receives response code 9 (ADARSP009).

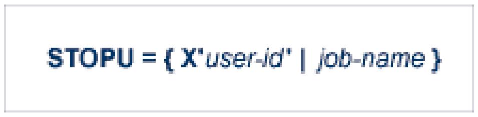
  This command does not stop EXF or UTI users.

  If the optional PURGE parameter is specified, the stopped users are also deleted (their user queue elements are removed from the user queue).

  The following is an example of using the PURGE parameter:

```
ADADBS OPERCOM STOPI=3600,PURGE
```

**Caution:**
If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPI command with PURGE only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur.

- 

  **STOPU = { X'***user-id***' |** *job-name* **}**

  Stop and delete the user with the Adabas-assigned user ID (in the form shown in the display commands), or stop and delete all users with the specified job name (*job-name*). Any open transaction by the stopped users will be backed out.

**Caution:**
If Adabas is running with ADARUN OPENRQ=NO (specifying that users are not required to issue an OP as the first command of the session), run the STOPU command only if you are certain that the users to be deleted are no longer active. If a user with an open transaction is deleted, but then returns (by sending a command), no indication is given about the transaction backout. If the user continues the transaction, logical inconsistencies in the database could occur.

**Note:**
The STOPU=X’*userid*’ command is not allowed for online reorder or invert processes. See the ONLSTOP=X’*identifier*’ command instead.

The user ID must be specified in hexadecimal format; for example:

```
STOPU=X’1CF2’
```

In a cluster environment, NUCID must always be specified because the user ID is not unique to the cluster.

- **SYNCC**

  Force resynchronization of all ET users on the nucleus. The nucleus waits for all ET users to reach ET status before continuing.

- **TNA***u* **=** *time*

  Set non-activity time limit (in seconds) for users where *u* is one of the following:

  - A - for access-only (ACC) users
  - E - for ET logic users
  - X - for exclusive control (EXF/EXU) users

  If specified, *time* must be a value greater than zero; it overrides the ADARUN value.

  In nucleus cluster environments, the TNAu commands are global by definition and affect all nuclei in the cluster. If a NUCID is specified, it is ignored.

- **TT =** *time*

Set transaction time limit (in seconds) for ET logic users. If specified, this value must be greater than zero; it overrides the ADARUN value. In nucleus cluster environments, the TT command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.

- **UNLOCKF =** *file-number*

  Unlock the specified file and restore its usage to the prelocked status.

- **UNLOCKU =** *file-number*

  Unlock the specified file for utility use and restore it to its prelocked status for non-utility users.

- **UNLOCKX =** *file-number*

  Unlock the specified file and restore its usage to the prelocked status.

- **UTIONLY = { YES | NO }**

  **Note:**
  Not currently available for use with Adabas Parallel Services cluster nuclei.

  Switch UTIONLY status on or off.

  In nucleus cluster environments, the UTIONLY command is global by definition and affects all nuclei in the cluster. If a NUCID is specified, it is ignored.