

Adabas Parallel Services

Adabas Parallel Services Operations

Version 7.5.1

September 2009

This document applies to Adabas Parallel Services Version 7.5.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © Software AG 2009. All rights reserved.

The name Software AG, webMethods and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

1 Adabas Parallel Services Operations	1
2 Initialization	3
Sequence	4
ADACOM Initialization Process	4
Adabas Parallel Services Cluster Nucleus Process	5
PPT Processing	6
3 Termination	9
Normal Termination	10
Abnormal Termination	10
4 Backout Processing	13
5 Restart/Recovery Processing	15
Offline Recovery (Session Autorestart)	16
Online Recovery	16
Automatic Restart Management (ARM)	17
Archive Recovery	17
6 Utility Processing	19
ADADBS OPERCOM Commands	20
ADADBS REFRESHSTATS - Refresh Statistical Values	21
ADAICK PPTPRINT - Print/Dump Parallel Participant Table	21
ADARAI - Adabas Recovery Aid	22
ADAREP - Checkpoint Information Extended	22
ADAPLP IPLOGPRI - Print Sequential Intermediate Data Sets	22
ADARES CLCOPY - Copy/Merge Nucleus Cluster Command Logs	22
ADARES PLCOPY - Copy/Merge Nucleus Cluster Protection Logs	24
ADARES MERGE CLOG - Merge Nucleus Cluster Command Logs	33
ADARES BACKOUT and REGENERATE— Uniquely Identifying Checkpoints	35
ADASAV Processing Change	36
ADASAV RESTPLOG -- Uniquely Identifying Checkpoints	37
7 Cluster Nucleus Session End Statistics	39
General Nucleus Information	40
Input/Output Statistics	40
Command Statistics	41
User Statistics	43
Efficiency Statistics	43
Global Cache Statistics (Cluster Nucleus Only)	44
Global Lock Statistics (Cluster Nucleus Only)	49
Data Set Activity Statistics	55
8 Performance and Tuning	57
ADARUN Parameter Settings	58
Allocating Work Data Set Space	58
Using Close (CL) Commands	58
Deferred Cache Operations	59

Tuning Buffer Flushes	60
Optimizing Global Cache and Lock Areas	60
Minimizing Communication with Global Areas	62
Optimizing Block Sizes	63
Index	65

1 Adabas Parallel Services Operations

This document provides information about initialization, termination, backout, restart, and recovery processes in an Adabas Parallel Services cluster environment. It tells you how to plan an outage and how to use the utility functions that are provided specifically for cluster environments. Finally, it provides a sample of the session-end statistics produced for a clustered nucleus and describes how to improve performance and tune Adabas Parallel Services.

This document is organized as follows:

• <i>Initialization</i>	Describes topics related to Adabas Parallel Services initialization.
• <i>Termination</i>	Describes topics related to Adabas Parallel Services termination.
• <i>Backout Processing</i>	Describes Adabas Parallel Services backout processing.
• <i>Restart/Recovery Processing</i>	Describes Adabas Parallel Services restart and recovery processing.
• <i>Utility Processing</i>	Describes Adabas Utilities processing pertinent to Adabas Parallel Services.
• <i>Cluster Nucleus Session End Statistics</i>	Describes the statistics collected for a cluster nucleus session.
• <i>Performance and Tuning</i>	Describes performance and tuning you can do for Adabas Parallel Services.

2 Initialization

- Sequence 4
- ADACOM Initialization Process 4
- Adabas Parallel Services Cluster Nucleus Process 5
- PPT Processing 6

This chapter covers the following topics related to Adabas Parallel Services initialization:

Sequence

Due to the interdependence among ADACOM(s) and Adabas cluster nuclei, these programs have certain co-occurrence requirements.

- ADACOM must be running on the operating system before any nucleus in a cluster that it manages is started.
- A nucleus will not start if another nucleus with the same NUCID is already active under the same router (SVC or IDTNAME).

ADACOM Initialization Process

The ADACOM initialization process is recorded in `PLI nnn` messages.

All `PLI nnn` messages are printed to the console. `PLI001-049` messages are specific to a particular Router ID / DBID set and are written to the output data set/file specifically allocated for that set; starting with `PLI050`, the messages apply to ADACOM in general and are written to the `COMPRINT` data set.



Note: Prior to `PLI050`, the system default ADARUN parameter values for `MODE`, `DBID`, `DEVICE`, `SVC` or `IDTNAME`, and `AMODE` are displayed. These are not the values for the current ADACOM.

```
PLI050 00161  INITIALIZING ADACOM
PLI002 00161  INITIALIZING DBID=dbid SVC=svc
                ACQUIRING NEW PLXCB
                PLXCB IS LOCATED AT address
PLI063 00161  PROCESSING: ADACOM SVC=svc,DBID=dbid,NU=users
                INITIALIZATION COMPLETE
```

A new Adabas Parallel Services control block is acquired if none yet exists.

Adabas Parallel Services Cluster Nucleus Process

Each Adabas Parallel Services cluster nucleus serializes during initialization by means of an enqueue. This is done because each nucleus puts information into the CSA nucleus table.

```

PLX050 00006 ADACLU init DBID=00006 NUCID=06001
PLX003 00006 There are nucs/ADACOMs active
PLX006 00006 Max users for image 00000200
PLX006 00006 PLXCB is located at 1127E258
ADAN03 00006 Initializing NUCID=6001 INTNUCID=4
ADAX2I 00006 SS-2,memstate, id 06001 ind 00 ext 00000      03
ADAX2A 00006 TI-0, initialised, rc 00000004
PLX087 00006 Attempting to acquire dataspace
PLX097 00006 Dataspace acquired
PLX059 00006 Initialization of ADACLU complete
ADAL01 00006 2002-06-25 17:50:45 CLOG not active
ADAN03 00006 ADABAS coming up
ADAX31 00006 Opening Work data set for NUCID=6001
ADAX70 00006 Connected to lock structure
ADAX70 00006 Number of lock entries          39,608
ADAX70 00006 Max number of record elements  39,608
ADAX57 00006 Connected to cache structure
ADAX57 00006 Directory elements          60,493
ADAX57 00006 Data elements          15,123
ADAX57 00006 Data element size          1,024
ADAN19 00006 Buffer flush is asynchronous
ADAN01 00006 ADABAS Vv.r.s is active
ADAN01 00006 MODE = MULTI
ADAN01 00006 Running without recovery log

```

Access to the global lock area is established and acknowledged in the following message:

```
ADAX70 00006 Connected to lock structure
```

This message is followed by lock area statistics:

```

ADAX70 00006 Number of lock entries          32,768
ADAX70 00006 Max number of record elements  11,184

```

Access to the global cache area is established and acknowledged in the following message:

```
ADAX57 00006 Connected to cache structure
```

If the ADARUN parameter CLUCACHETYPE=V64 was specified, this message appears:

```
ADAX57 00006 Connecting to S64 cacache at address
```

This message is followed by cache area statistics:

```
ADAX57 00006 Directory elements      11,490
ADAX57 00006 Data elements           2,872
ADAX57 00006 Data element size      1,024
ADAN19 00006 Buffer flush is asynchronous
ADAN01 00006 ADABAS Vv.r.s is active
ADAN01 00006 Mode = MULTI
ADAN01 00006 Running without recovery log
```

PPT Processing

The parallel participant table (PPT), which exists for both cluster and noncluster nuclei, is used to determine if any PLOGs still need to be copied from previous sessions. If the PPT indicates that PLOGs remain to be copied, the PLOG data sets are read and, if necessary, a user exit is invoked (user exit 2 for dual logs or user exit 12 for multiple logs).

This section covers the following topics:

- [First Cluster Nucleus Starts](#)
- [Subsequent Cluster Nucleus Starts](#)
- [Noncluster Nucleus Starts](#)
- [Different PLOG Detected](#)

First Cluster Nucleus Starts

The Adabas Parallel Services cluster nucleus that is the first to initialize checks all the PLOG entries from the previous session for all nuclei and marks any that are "still being written" to completed status. In this way, the user exit (user exit 2 for dual logs or user exit 12 for multiple logs) need not be called each time a cluster nucleus autostarts. The first nucleus then calls the user exit but waits only if the PLOGs that need to be copied are for its own NUCID.

Subsequent Cluster Nucleus Starts

A subsequent cluster nucleus checks only its own PLOGs and invokes the user exit (user exit 2 for dual logs or user exit 12 for multiple logs) if the PLOGs are still not copied/merged. It waits if the user exit instructs it to do so. If there is no user exit 2 or 12, it overwrites the PLOGs.

Noncluster Nucleus Starts

A noncluster nucleus checks whether the previous session was an Adabas Parallel Services cluster session and has a pending autorestart. If so, the noncluster nucleus is not allowed to start.

If PLOGs from a previous Adabas Parallel Services cluster session remain to be copied, ADARES invokes the merge or the PLCOPY as required. A noncluster nucleus always uses block 1 of the PPT and can only overwrite it when PLOGs from previous sessions have been processed to completion.

A user exit (user exit 2 for dual logs or user exit 12 for multiple logs) controls the copy/merge process. If there is no user exit 2 or 12, the PLOG and PPT entry are overwritten.

Different PLOG Detected

If PLOGRQ=FORCE and an uncopied PLOG is detected that does not match that specified in the last session, a parameter error occurs. If the PLOG has been copied, the PPT entry is overwritten and the nucleus starts.

3 Termination

- Normal Termination 10
- Abnormal Termination 10

This chapter covers the following topics related to Adabas Parallel Services termination:

Normal Termination

This section covers the following topics:

- [ADACOM](#)
- [Cluster Nuclei](#)

ADACOM

ADACOM must stay in operation in an active Adabas Parallel Services cluster environment. ADACOM will not terminate normally if any nucleus that it manages is still active.

When all nuclei that it manages have terminated, you can terminate ADACOM using an ADAEND command. See the section *Cluster Operator Commands* in the *Adabas Parallel Services Reference Guide*.

Cluster Nuclei

If the Adabas operator command ADAEND or HALT is issued, the nucleus stops with no pending autorestart. The other active nuclei in the cluster continue processing normally.

```
ADAN51 00006 2002-06-25 18:03:29 Operator type-in: ADAEND
ADAN42 00006 2002-06-25 18:03:29 Function accepted
ADAX2B 00006 TT-1, SMM terminating
PLX087 00006 Attempting to delete dataspaces
PLX092 00006 Dataspaces deleted
ADAM97 06001 This ASCB/Initiator will be terminated by MVS at E0J
```

If the ADARUN parameter CLUCACHETYPE=V64 was specified, this message also appears:

```
ADAX5B 00006 Disconnecting from S64 cache
```

Abnormal Termination

This section covers the following topics:

- [ADACOM](#)

- Cluster Nuclei

ADACOM

If ADACOM terminates abnormally, a `PLInnn` error message is produced to explain the problem. All active Parallel Services nuclei in clusters managed by this instance of ADACOM will also abend. ADACOM is the owner of the cache and lock data spaces used by the cluster nuclei. If ADACOM goes away, the data spaces also go away, and the nuclei will abend when they attempt to continue accessing the data spaces.



Note: This is not true of Adabas Cluster Services, where the cache and lock structures exist until all connectors (Adabas nuclei) terminate.

Cluster Nuclei

When an Adabas Parallel Services cluster nucleus terminates abnormally, each surviving peer nucleus performs online recovery. Read [Restart/Recovery](#) elsewhere in this guide for more information.

4 Backout Processing

Normal backout processing includes:

- BT command processing;
- backing out an update command that received a nonzero response code; and
- internal transaction backout due to, for example, a timeout.

Cluster nuclei perform normal Adabas backout processing. Each nucleus reads the protection data needed to back out updates from its own Work data set/file. It does not need to read protection data from the other Work data sets/files in the cluster.

5 Restart/Recovery Processing

- Offline Recovery (Session Autorestart) 16
- Online Recovery 16
- Automatic Restart Management (ARM) 17
- Archive Recovery 17

Restart/recovery occurs if a cluster nucleus fails. Restart/recovery uses the Work data sets/files of all nuclei to recover the database. The Work data sets/files are dynamically allocated from the data set names recorded in the PPT. Adabas Parallel Services version supports offline and online recovery.

This chapter covers the following topics:

Offline Recovery (Session Autorestart)

- If a cluster nucleus session terminates abnormally, start one of the cluster nuclei to perform the autorestart.
- If a noncluster nucleus session terminates abnormally, restart the noncluster nucleus to perform the autorestart.

Offline recovery occurs if all active cluster nuclei in an Adabas Parallel Services cluster fail. Offline recovery relies only on information from the physical database and the Work data sets/files of each cluster nucleus. All information in the global cache and lock areas is lost.

The first cluster nucleus to restart repairs any physical inconsistencies in the database and backs out all incomplete commands and transactions. The restarted nucleus obtains recovery information from blocks in the common database and from the Work data sets/files of all the failed nuclei.

The restarting nucleus retrieves the Work data set/file names from the PPT block for each terminated nucleus and opens these data sets/files using dynamic allocation. From that point, normal recovery processing occurs:

- the breakpoint on each Work data set/file is found;
- backward and forward repair is performed; and
- autobackout is performed.

While reading through the Work data sets/files, the restarting nucleus on the fly merges the protection records by their timestamps into chronological sequence.

Online Recovery

When one or more cluster nuclei have failed while one or more other nuclei in the same cluster remain active, online recovery processing is performed by collaboration of all surviving nuclei.

All surviving cluster nuclei quiesce their operations and reinitialize their working storage. Command processing is quiesced and the internal status variables, tables, and pools are repaired.

The peer nuclei compete for the recovery lock: when one of the nuclei obtains it, it invokes offline recovery processing. It repairs any physical inconsistencies in the database and backs out all incomplete command and transactions. Open transactions executed by the surviving nuclei are backed out as well. All information in the global lock and cache areas is discarded.

Once this recovery processing has completed, normal processing resumes.

Users are affected by online recovery as follows:

- users assigned to failed nuclei lose their commands, transactions, sequential processes, and search results. They may receive response codes 9, 21, 148, or 251, depending on the status of their session at the time of the failure.
- users assigned to surviving nuclei may or may not lose their commands/transactions, depending on whether they managed to complete them in the quiesce phase. They retain their sequential processes and search results, but they may experience an increased response time. Users that do lose their commands/transactions will subsequently receive response code 9 and might possibly get response code 21 as well.

Automatic Restart Management (ARM)

Automatic restart management (ARM) is an OS/390 and z/OS facility that can be used to automatically restart a nucleus when it ABENDs. Automatic restart is suppressed when the ABEND is intentional; for example, when it results from a parameter error.

ARM can be used for Adabas nuclei in both cluster and noncluster environments.

The ADARUN parameter `ARMNAME` (read *Parameter Directory* in the *Adabas Parallel Services Reference Guide*) is used to identify the element in the ARM *policy* that is to be activated. Each element specifies when, where, and how often an automatic restart is to be attempted.

If an ARM policy has not been defined, the `ARMNAME` parameter has no effect.

Archive Recovery

Archive recovery occurs if the container data sets of the database are damaged or restart recovery is not effective.

Archive recovery:

- restores the database; and
- regenerates the updates from the protection logs.

The protection logs to be regenerated are the output of the ADARES PLCOPY protection log copy and merge process that occurs in Adabas Parallel Services cluster environments. The restore/regenerate process is the same in both cluster and noncluster environments.

6 Utility Processing

▪ ADADBS OPERCOM Commands	20
▪ ADADBS REFRESHSTATS - Refresh Statistical Values	21
▪ ADAICK PPTPRINT - Print/Dump Parallel Participant Table	21
▪ ADARAI - Adabas Recovery Aid	22
▪ ADAREP - Checkpoint Information Extended	22
▪ ADAPLP IPLOGPRI - Print Sequential Intermediate Data Sets	22
▪ ADARES CLCOPY - Copy/Merge Nucleus Cluster Command Logs	22
▪ ADARES PLCOPY - Copy/Merge Nucleus Cluster Protection Logs	24
▪ ADARES MERGE CLOG - Merge Nucleus Cluster Command Logs	33
▪ ADARES BACKOUT and REGENERATE— Uniquely Identifying Checkpoints	35
▪ ADASAV Processing Change	36
▪ ADASAV RESTPLOG -- Uniquely Identifying Checkpoints	37

Read your *Adabas Utilities Manual* for specific information about changes to utilities for use in an Adabas cluster environment.

This chapter covers the following topics:

ADADBS OPERCOM Commands

Changes have been made for ADADBS OPERCOM command processing in an Adabas Parallel Services cluster nucleus environment.

This section covers the following topics:

- [Global Commands](#)
- [Routing a Command to a Specific Nucleus](#)
- [Routing a Command to All Cluster Nuclei](#)

Global Commands

The following ADADBS OPERCOM commands have a GLOBAL option for making the request to all nuclei in an Adabas cluster:

- ADAEND
- CANCEL
- FEOFCL
- FEOFPL
- HALT

For example:

```
ADADBS OPERCOM ADAEND,GLOBAL
```

When GLOBAL is specified, the command is automatically propagated to all active cluster nuclei. When GLOBAL is *not* specified, a specific NUCID from the cluster must be specified and the command is sent to that NUCID.

Routing a Command to a Specific Nucleus

The NUCID option allows you to direct the OPERCOM commands to a particular nucleus in the cluster for execution.

The OPERCOM function's NUCID option is specified in a manner similar to the ADARUN function's NUCID parameter.

The following example sends the DSTAT command to the Adabas cluster nucleus designated with NUCID=3:

```
ADADBS OPERCOM DSTAT,NUCID=3
```

For inherently global commands, such as changing the setting of the TT parameter, the NUCID parameter is ignored.

Routing a Command to All Cluster Nuclei

When the NUCID option in the ADADBS OPERCOM function is not specified, the command is sent to all cluster nuclei and information is displayed for each nucleus in sequence.

ADADBS REFRESHSTATS - Refresh Statistical Values

The REFRESHSTATS function resets statistical values maintained by the Adabas nucleus for its current session. Parameters may be used to restrict the function to particular groups of statistical values.

In Adabas cluster environments, you must specify the specific nucleus (NUCID) for which statistical values are to be refreshed. If NUCID is not specified, statistical values will be refreshed for all active nuclei in the cluster.

ADAICK PTPRINT - Print/Dump Parallel Participant Table

The PTPRINT function has been added to the Adabas ADAICK utility to support an Adabas cluster environment. It is used to dump/print the parallel participant table (PPT) for the Adabas cluster.

Each of the 32 blocks (RABNs) allocated for the PPT represents a single nucleus in the cluster and comprises:

- a single header of fixed length; and
- multiple entries of variable length.

Note that in the dump/print, "PPH" is the tag for the PPT header and "PPE" is the tag for the PPT entries.

ADARAI - Adabas Recovery Aid

Adabas cluster products support the Adabas Recovery Aid (ADARAI).

ADARAI maintains a recovery log (RLOG) for each database; all nuclei in the cluster support a database write to the same RLOG and concurrent updates to the RLOG are controlled by a lock.

The ADARAI LIST function supports Adabas version 7 and above RLOGs; Adabas version 6 RLOGs are not supported.

ADAREP - Checkpoint Information Extended

Given that each cluster nucleus has its own PLOG data sets, checkpoints are no longer identified only by their name, PLOG number, and PLOG block number, but also by the ID of the nucleus that writes the checkpoint.

Several new parameters have been introduced for utilities that need to identify checkpoints on the PLOG.

ADAPLP IPLOGPRI - Print Sequential Intermediate Data Sets

The IPLOGPRI function is used to print the sequential intermediate data sets created from the PLOG merge process. Input to ADAPLP IPLOGPRI must be a MERGIN1/MERGIN2 data set created by the ADARES utility and specified in the JCL with DD name/link name DD/PLOG.

ADARES CLCOPY - Copy/Merge Nucleus Cluster Command Logs

Sample JCL has been added for allocating the intermediate data sets/files MERGIN1 and MERGIN2 required for automated CLOG copy/merge processing in nucleus cluster environments.



Notes:

1. When intermediate data sets/files are used for both CLCOPY and PLCOPY, the data set names must be unique so that they are not overwritten.

2. The data set BLKSIZE used must be greater than or equal to the largest CLOG BLKSIZE plus eight. The LRECL must be set to the BLKSIZE minus four.

- OS/390 and z/OS
- VSE
- BS2000/OSD

OS/390 and z/OS

```
//ALLOC JOB
//*
/* Example to allocate the ADARES CLCOPY intermediate data sets
/*
//CM1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//OUTFIL DD DISP=(NEW,CATLG),SPACE=(CYL,(3,10)),UNIT=SYSDA,
// VOL=SER=volser,DCB=(RECFM=VB,LRECL=23472,BLKSIZE=23476),
// DSN=EXAMPLE.CLOG.MERGIN1
//INPFIL DD *
/*
//SYSIN DD *
REPRO INFILE(INPFIL) -
OUTFILE(OUTFIL)
/*
/*
//CM2 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//OUTFIL DD DISP=(NEW,CATLG),SPACE=(CYL,(3,10)),UNIT=SYSDA,
// VOL=SER=volser,DCB=(RECFM=VB,LRECL=23472,BLKSIZE=23476),
// DSN=EXAMPLE.CLOG.MERGIN2
//INPFIL DD *
/*
//SYSIN DD *
REPRO INFILE(INPFIL) -
OUTFILE(OUTFIL)
/*
```

VSE

```
//JOB ALLOC
/* Example to allocate the ADARES CLCOPY intermediate data sets
//EXEC PGM=IDCAMS
DEFINE CLUSTER (NAME (MERGIN1.CLCOPY) -
VOLUME(xxxxxx) CYLINDERS(3,10) RECSZ(23472) -
/*
//EXEC PGM=IDCAMS
DEFINE CLUSTER (NAME (MERGIN2.CLCOPY) -
VOLUME(xxxxxx) CYLINDERS(3,10) RECSZ(23472) -
```

```
/*  
/&
```

BS2000/OSD

```
/BEGIN PROC A  
/REMARK Example to allocate the ADARES CLCOPY intermediate files  
/CREATE-FILE ADAddd.CLOG.MERGIN1,PUB(SPACE=(96,960))  
/SET-FILE-LINK MERGIN1,ADAddd.CLOG.MERGIN1,BLKSIZE=STD(14)  
/CREATE-FILE ADAddd.CLOG.MERGIN2,PUB(SPACE=(96,960))  
/SET-FILE-LINK MERGIN2,ADAddd.CLOG.MERGIN2,BLKSIZE=STD(14)  
.  
.  
/END-PROC
```

ADARES PLCOPY - Copy/Merge Nucleus Cluster Protection Logs

In an Adabas nucleus cluster environment, the protection logs (and optionally, the command logs: see the ADARES MERGE CLOG function and the ADARUN CLOGMRG parameter) of all individual nuclei in the cluster must be merged into single log files in chronological order for the cluster database shared by all the nuclei as a whole. The chronological order is determined by timestamps on all individual nucleus log records.

Protection logs are automatically merged when an ADARES PLCOPY is executed. In an Adabas cluster environment, the PLCOPY process accesses the parallel participant table (PPT) to determine which protection logs to copy and opens the appropriate data sets/files using dynamic allocation. PLCOPY copies/merges as much data as possible; if a nucleus is still writing to a protection log data set, PLCOPY 'partially' merges the data set.

The merge begins with the lowest timestamp from all protection logs being merged and ends with the lowest of the ending timestamps from all data sets/files. Records beyond this point are written to an 'intermediate' data set, which must be supplied as input to the subsequent merge. A cross-check ensures that the correct intermediate data set has been supplied.

ADARES expects that at least one of the protection logs being merged is at 'completed' status. If this is not the case, ADARES reports that there is no data to be copied.

A sample user exit (USEREX2P for dual logs or UEX12 for multiple logs) is provided to illustrate the necessary change for the intermediate data set.

- [OS/390 and z/OS Sample Jobs](#)
- [VSE/ESA Sample Jobs](#)

- [BS2000 Sample Jobs](#)

OS/390 and z/OS Sample Jobs

This section describes the sample jobs provided for OS/390 and z/OS systems.

ALLOC Job

The following sample JCL illustrates the allocation of the intermediate data sets MERGIN1 and MERGIN2 which are required for automated PLOG copy/merge processing in nucleus cluster environments.



Notes:

1. When intermediate data sets are used for both CLCOPY and PLCOPY, the data set names must be unique so that they are not overwritten.
2. The data set BLKSIZE used must be greater than or equal to the largest PLOG BLKSIZE plus eight. The LRECL must be set to the BLKSIZE minus four.

```
//ALLOC JOB
//*
//* Example to allocate the ADARES PLCOPY intermediate data sets
//*
//CM1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//OUTFIL DD DISP=(NEW,CATLG),SPACE=(CYL,(3,10)),UNIT=SYSDA,
// VOL=SER=volser,DCB=(RECFM=VB,LRECL=23472,BLKSIZE=23476),
// DSN=EXAMPLE.PLOG.MERGIN1
//INPFIL DD *
/*
//SYSIN DD *
REPRO INFILE(INPFIL) -
OUTFILE(OUTFIL)
/*
//*
//CM2 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//OUTFIL DD DISP=(NEW,CATLG),SPACE=(CYL,(3,10)),UNIT=SYSDA,
// VOL=SER=volser,DCB=(RECFM=VB,LRECL=23472,BLKSIZE=23476),
// DSN=EXAMPLE.PLOG.MERGIN2
//INPFIL DD *
/*
//SYSIN DD *
REPRO INFILE(INPFIL) -
OUTFILE(OUTFIL)
/*
```

ADARESPM Job

A sample job ADARESPM is provided on the JOBS data set to illustrate the manual execution of the PLCOPY merge function. Two intermediate data sets must be supplied. ADARES analyzes the data sets to determine which is to be used as input and which for output. Specific crosschecks determine whether the correct intermediate data set has been supplied; if not, ADARES will not continue. Continuing without the correct input can result in lost updates and inconsistencies if the output is used for REGENERATE or BACKOUT functions.

Once DDPLOGRn statements have been supplied on the session start-up JCL, you do not need to supply them again for ADARES as these are opened using dynamic allocation. If the DD statements are supplied, they are ignored.

The following sample JCL illustrates the ADARES PLCOPY merge function:

```
//ADARESPM JOB
//*
//* ADARES : COPY/MERGE DUAL/MULTIPLE PROTECTION LOG
//* TWO COPIES OF OUTPUT ARE TO BE CREATED
//* FOR USE WITH AN ADABAS NUCLEUS CLUSTER
//*
//RES EXEC PGM=ADARUN
//STEPLIB DD DISP=SHR,DSN=ADABAS.Vvrs.LOAD
//*
//DDASSOR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1
//DDDATAR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1
//MARGIN1 DD DISP=SHR,DSN=EXAMPLE.PLOG.MARGIN1
//MARGIN2 DD DISP=SHR,DSN=EXAMPLE.PLOG.MARGIN2
//DDSIAUS1 DD DSN=EXAMPLE.DByyyyy.PLOG1(+1),
// VOL=SER=ADAxxx,UNIT=TAPE,DISP=(NEW,CATLG)
//DDSIAUS2 DD DSN=EXAMPLE.DByyyyy.PLOG2(+1),
// VOL=SER=ADAxxx,UNIT=TAPE,DISP=(NEW,CATLG)
//DDDRUCK DD SYSOUT=X
//DDPRINT DD SYSOUT=X
//SYSUDUMP DD SYSOUT=X
//DDCARD DD *
ADARUN PROG=ADARES,MODE=MULTI,SVC=svc,DEVICE=3380,DBID=yyyyy
/*
//DDKARTE DD *
ADARES PLCOPY TWOCOPIES
/*
```

ADARES PLCOPY NOPPT—Ignore PPT


NOPPT is for emergency use when the PPT has been overwritten or is otherwise unavailable. It specifies that the PLOG data sets of all cluster nuclei are being supplied with DD names DDPLOGnn in the JCL.

-  **Caution:** Use this parameter cautiously since it ignores the PPT and all control-type information typically coming from the PPT.

When you use this parameter, you must supply

- the correct intermediate data set; and
- the correct input protection logs from all nuclei with DD names DDPLOG01-nn.

The optional parameter `SBLKNUM` can be used to specify the starting block number for the sequential merge output.

-  **Caution:** Without the PPT, ADARES cannot perform any extensive validations on the input data sets

ADARESIP Job

The following sample JCL illustrates the ADARES PLCOPY NOPPT merge function:

```
//ADARESIP JOB
//*
//* ADARES : COPY/MERGE DUAL/MULTIPLE PROTECTION LOGS FROM ALL
//* NUCLEI IN AN ADABAS NUCLEUS CLUSTER
//* PPT IS TO BE IGNORED
//* THIS IS ONLY FOR EMERGENCY USE WHEN THE PPT HAS BEEN
//* OVER-WRITTEN - USE CAUTION WHEN SUBMITTING
//*
//RES EXEC PGM=ADARUN
//STEPLIB DD DISP=SHR,DSN=ADABAS.Vvrs.LOAD <=== ADABAS LOAD
//*
//DDASSOR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDDATAR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDPLOG01 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1.NUC1 <=== PLOG1 NUC1
//DDPLOG02 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR2.NUC1 <=== PLOG2 NUC1
//DDPLOG03 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1.NUC2 <=== PLOG1 NUC2
//DDPLOG04 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR2.NUC2 <=== PLOG2 NUC2
//DDPLOG05 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR1.NUC3 <=== PLOG1 NUC3
//DDPLOG06 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.PLOGR2.NUC3 <=== PLOG2 NUC3
//MARGIN1 DD DISP=SHR,DSN=EXAMPLE.PLOG.MARGIN1 <=== INTERMEDIATE 1
//MARGIN2 DD DISP=SHR,DSN=EXAMPLE.PLOG.MARGIN2 <=== INTERMEDIATE 2
//DDSIAUS1 DD DSN=EXAMPLE.DByyyyy.PLOG1(+1), <=== PLOG COPY
// VOL=SER=ADAxxx,UNIT=TAPE,DISP=(NEW,CATLG)
//DDDRUCK DD SYSOUT=X
//DDPRINT DD SYSOUT=X
//SYSUDUMP DD SYSOUT=X
//DDCARD DD *
ADARUN PROG=ADARES,MODE=MULTI,SVC=svc,DEVICE=3380,DBID=yyyyy
/*
//DDKARTE DD *
ADARES PLCOPY NOPPT
```

```
/*  
//
```

VSE/ESA Sample Jobs

This section describes the sample jobs provided for VSE/ESA systems.

ALLOC Job

The following sample JCL is provided for allocating the intermediate data sets MERGIN1 and MERGIN2 which are required for automated PLOG copy/merge processing in nucleus cluster environments.



Notes:

1. When intermediate data sets are used for both CLCOPY and PLCOPY, the data set names must be unique so that they are not overwritten.
2. The data set block size used must be greater than or equal to the largest PLOG block size plus eight. The record length must be set to the block size minus four.

```
// JOB MRGFILE  
// ASSGN SYS004,SYSIPT  
// ASSGN SYS005,DISK,VOL-VSE209,SHR  
// DLBL UOUT,'QA.DB1010.MERGIN1',0  
// EXTENT SYS005,VSE209,1,0,10575,300  
// EXEC PGM=OBJMAINT  
./ CARD DLM=$$  
./ COPY  
$$  
/*  
// ASSGN SYS004,SYSIPT  
// ASSGN SYS005,DISK,VOL=VSE209,SHR  
// DLBL UOUT,'QA.DB1010.MERGIN2',0  
// EXTENT SYS005,VSE209,1,0,11625,300  
// EXEC PGM=OBJMAINT  
./ CARD DLM=$$  
./ COPY  
$$  
/*  
/&
```

ADARESPM Job

A sample job ADARESPM is provided to illustrate the manual execution of the PLCOPY merge function. Two intermediate data sets must be supplied. ADARES analyzes the data sets to determine which is to be used as input and which for output. Specific crosschecks determine whether the correct intermediate data set has been supplied; if not, ADARES will not continue. Continuing

without the correct input can result in lost updates and inconsistencies if the output is used for REGENERATE or BACKOUT functions.

Once PLOGRn statements have been supplied on the session start-up JCL, you do not need to supply them again for ADARES as these are opened using dynamic allocation. If the PLOGRn statements are supplied, they are ignored.

The following sample JCL illustrates the ADARES PLCOPY merge function:

```
// JOB ADARESPM
// *
// * ADARES : COPY/MERGE DUAL/MULTIPLE PROTECTION LOG
// * TWO COPIES OF OUTPUT ARE TO BE CREATED
// * FOR USE WITH AN ADABAS NUCLEUS CLUSTER
// *
// LIBDEF PHASE,SEARCH=(SAGLIB.ASM742,SAGLIB.ADA742)
// DLBL ASSOR1,'EXAMPLE.DByyyyy.ASSOR1'
// DLBL DATAR1,'EXAMPLE.DByyyyy.DATAR1'
// DLBL MERGIN1,'MERGIN1.PLCOPY'
// DLBL MERGIN2,'MERGIN2.PLCOPY'
// TLBL SIAUS1,'EXAMPLE.DByyyyy.PLOG1'
// TLBL SIAUS2,'EXAMPLE.DByyyyy.PLOG2'
// ASSGN SYS021,TAPE
// EXEC ADARUN,SIZE=ADARUN
PROG=ADARES,MODE=MULTI,SVC=svc,DEVICE=3380,DBID=yyy
yy
/*
ADARES PLCOPY TWOCOPIES
/*
/&
```


ADARES PLCOPY NOPPT—Ignore PPT

NOPPT is for emergency use when the PPT has been overwritten or is otherwise unavailable. It specifies that the PPT is to be ignored and that the PLOG data sets of all cluster nuclei are being supplied with link names PLOGnn in the JCL.

 **Caution:** Use this parameter cautiously since it ignores the PPT and all control-type information typically coming from the PPT.

When you use this parameter, you must supply

- the correct intermediate data set; and
- the correct input protection logs from all nuclei with link names PLOG01-nn.

 **Caution:** Without the PPT, ADARES cannot perform any extensive validations on the input data sets.

ADARESIP Job

The following sample job ADARESIP is used with the ADARES PLCOPY NOPPT function:

```
// JOB ADARESIP
// *
// * ADARES : COPY/MERGE DUAL/MULTIPLE PROTECTION LOG
// * FOR USE WITH AN ADABAS NUCLEUS CLUSTER
// * CAUTION: NOPPT EXECUTION!
// LIBDEF PHASE,SEARCH=(SAGLIB.ASM742,SAGLIB.ADA742)
// DLBL ASSOR1,'EXAMPLE.DByyyyy.ASSOR1'
// DLBL DATAR1,'EXAMPLE.DByyyyy.DATAR1'
// DLBL PLOG01,'EXAMPLE.DByyyyy.PLOGR1.NUC1'
// DLBL PLOG02,'EXAMPLE.DByyyyy.PLOGR2.NUC1'
// DLBL PLOG03,'EXAMPLE.DByyyyy.PLOGR1.NUC2'
// DLBL PLOG04,'EXAMPLE.DByyyyy.PLOGR2.NUC2'
// DLBL PLOG05,'EXAMPLE.DByyyyy.PLOGR1.NUC3'
// DLBL PLOG06,'EXAMPLE.DByyyyy.PLOGR2.NUC3'
// DLBL MERGIN1,'MERGIN1.PLCOPY'
// DLBL MERGIN2,'MERGIN2.PLCOPY'
// TLBL SIAUS1,'EXAMPLE.DByyyyy.PLOG1'
// ASSGN SYS021,TAPE
// EXEC ADARUN,SIZE=ADARUN
PROG=ADARES,MODE=MULTI,SVC=svc,DEVICE=3380,DBID=yyyyy
/*
ADARES PLCOPY NOPPT
/*
/&
```

BS2000 Sample Jobs

This section describes the sample jobs provided for BS2000 systems.

ALLOC Job

For BS2000 systems, sample JCL is provided for allocating the intermediate files MERGIN1 and MERGIN2, which are required for automated PLOG copy/merge processing in nucleus cluster environments.



Notes:

1. When intermediate files are used for both CLCOPY and PLCOPY, the file names must be unique so that they are not overwritten.
2. The file BLKSIZE used must be greater than or equal to the largest PLOG BLKSIZE plus eight. The LRECL must be set to the BLKSIZE minus four.

```

/BEGIN PROC A
/REMARK Example to allocate the ADARES PLCOPY intermediate files
/CREATE-FILE ADAddd.PLOG.MERGIN1,PUB(SPACE=(96,960))
/SET-FILE-LINK MERGIN1,ADAddd.PLOG.MERGIN1,BLKSIZE=STD(14)
/CREATE-FILE ADAddd.PLOG.MERGIN2,PUB(SPACE=(96,960))
/SET-FILE-LINK MERGIN2,ADAddd.PLOG.MERGIN2,BLKSIZE=STD(14)
.
.
/END-PROC

```

ADARESPM Job

A sample job ADARESPM is provided on the JOBS file to illustrate the manual execution of the PLCOPY merge function. Two intermediate files must be supplied. ADARES analyzes the files to determine which is to be used as input and which for output. Specific crosschecks determine whether the correct intermediate file has been supplied; if not, ADARES will not continue. Continuing without the correct input can result in lost updates and inconsistencies if the output is used for REGENERATE or BACKOUT functions.

Once DDPLOGRn statements have been supplied on the session start-up JCL, you do not need to supply them again for ADARES as these are opened using dynamic allocation. If the file link statements are supplied, they are ignored.

The following sample JCL illustrates the ADARES PLCOPY merge function:

```

/BEGIN-PROC C
/MOD-TEST DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY/MERGE DUAL PROTECTION LOG
/REMARK * 2 COPIES OF OUTPUT ARE TO BE CREATED
/REMARK * FOR USE IN A PARALLEL SERVICES
/REMARK * DATABASE
/REMARK *
/DEL-FI ADAddd.AUS1
/SET-JOB-STEP
/CREATE-FILE ADAddd.AUS1,SUP=TAPE(VOL=RES101,DEV-TYPE=T-C4),-
/ PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/DEL-FI ADAddd.AUS2
/SET-JOB-STEP
/CREATE-FILE ADAddd.AUS2,SUP=TAPE(VOL=RES102,DEV-TYPE=T-C4),-
/ PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/ASS-SYSLST L.RES.PAR
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADABAS.MOD
/SET-FILE-LINK BLSLIB00,ADAASM.MOD
/SET-FILE-LINK DDASSOR1,ADAddd.ASSO ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDDATAR1,ADAddd.DATA ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK MERGIN1,ADAddd.PLOG.MERGIN1 ,SUP=DISK(SHARE-UPD=YES)

```

```

/SET-FILE-LINK MERGIN2,ADAddd.PLOG.MERGIN2 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDSIAUS1,ADAddd.AUS1,ACC-METHOD=SAM, -
/ BUFF-LEN=32768,REC-FORM=V,SUP=TAPE(LABEL=STD)
/SET-FILE-LINK DDSIAUS2,ADAddd.AUS2,ACC-METHOD=SAM, -
/ BUFF-LEN=32768,REC-FORM=V,SUP=TAPE(LABEL=STD)
/START-PROG *(E=ADARUN,L=ADABAS.MOD),RUN-MODE=ADV(A-L=YES)
ADARUN PROG=ADARES,DB=ddd
ADARES PLCOPY TWOCOPIES
/ASS-SYSDTA *PRIM
/ASS-SYSLST *PRIM
/END-PROC

```

ADARESIP Job

The following sample JCL illustrates the ADARES PLCOPY NOPPT merge function:

```

/BEGIN-PROC C
/MOD-TEST DUMP=YES
/REMARK *
/REMARK * A D A R E S COPY/MERGE DUAL PROTECTION LOGS
/REMARK * IN A PARALLEL SERVICES NUCLEUS
/REMARK * PPT IS TO BE IGNORED
/REMARK * THIS IS AN EMERGENCY USE FOR WHEN PPT
/REMARK * HAS BEEN OVERWRITTEN - USE CAUTION
/DEL-FI ADAddd.AUS
/SET-JOB-STEP
/CREATE-FILE ADAddd.AUS,SUP=TAPE(VOL=RES101,DEV-TYPE=T-C4), -
/ PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/ASS-SYSLST L.RES.DELT
/ASS-SYSDTA *SYSCMD
/DEL-FI ADAddd.AUS
/SET-JOB-STEP
/CREATE-FILE ADAddd.AUS,SUP=TAPE(VOL=RES103,DEV-TYPE=T-C4), -
/ PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/SET-FILE-LINK DDLIB,ADABAS.MOD
/SET-FILE-LINK BLSLIB00,ADAASM.MOD
/SET-FILE-LINK DDASSOR1,ADAddd.ASSO ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDDATAR1,ADAddd.DATA ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDWORKR1,ADAddd.WORK ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDPLOG01,ADAddd.PLOG1.NUC1 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDPLOG02,ADAddd.PLOG2.NUC1 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDPLOG03,ADAddd.PLOG1.NUC2 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDPLOG04,ADAddd.PLOG2.NUC2 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDPLOG05,ADAddd.PLOG1.NUC3 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDPLOG06,ADAddd.PLOG2.NUC3 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK MERGIN1,ADAddd.PLOG.MERGIN1,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK MERGIN2,ADAddd.PLOG.MERGIN2,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDSIAUS1,ADAddd.AUS,ACC-METHOD=SAM, -
/ BUFF-LEN=32768,REC-FORM=V,SUP=TAPE(LABEL=STD)

```

```

/START-PROG *M(E=ADARUN,L=ADABAS.MOD),RUN-MODE=ADV(A-L=YES)
ADARUN PROG=ADARES,DB=ddd
ADARES PLCOPY NOPPT
/ASS-SYSDTA *PRIM
/ASS-SYSLST *PRIM
/END-PROC

```

ADARES MERGE CLOG - Merge Nucleus Cluster Command Logs

In an Adabas cluster environment, command logs (CLOGs) from the cluster nuclei may be manually merged using the ADARES MERGE CLOG NUMLOG=nn function.

The NUMLOG parameter is required: it specifies the number of command log data sets/files to be included in the merge process. The maximum number is 32.

Sequential data sets/files are expected as input to the MERGE CLOG function; therefore, the ADARES CLCOPY function must be executed prior to the ADARES MERGE function.

The timestamp contained in the CLOGLAYOUT=5 format of the CLOG is required for the proper merging of command logs records.

- OS/390 and z/OS
- VSE/ESA
- BS2000

OS/390 and z/OS

The following sample job ADADESCM for OS/390 and z/OS systems (see the JOBS data set) illustrates the execution of the ADARES MERGE CLOG function:

```

//ADADESCM JOB
//*
//* ADARES : MERGE SEQUENTIAL COMMAND LOGS
//* FOR USE WITH AN ADABAS NUCLEUS CLUSTER
//*
//RES EXEC PGM=ADARUN
//STEPLIB DD DISP=SHR,DSN=ADABAS.Vvrs.LOAD <=== ADABAS LOAD
//*
//DDASSOR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.ASSOR1 <=== ASSO
//DDATAR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.DATAR1 <=== DATA
//DDWORKR1 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.WORKR1 <=== WORK
//DDCLOG01 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.CLOGR1.NUC1 <=== CLOG1 NUC1
//DDCLOG02 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.CLOGR1.NUC2 <=== CLOG1 NUC2
//DDCLOG03 DD DISP=SHR,DSN=EXAMPLE.DByyyyy.CLOGR2.NUC3 <=== CLOG2 NUC3
//DDSIAUS1 DD DSN=EXAMPLE.DByyyyy.CLOGM, <=== OUTPUT OF
// VOL=SER=ADAxxx,UNIT=TAPE,DISP=(NEW,CATLG) CLOG MERGE

```

```
//DDDRUCK DD SYSOUT=X
//DDPRINT DD SYSOUT=X
//SYSUDUMP DD SYSOUT=X
//DDCARD DD *
ADARUN PROG=ADARES,MODE=MULTI,SVC=svc,DEVICE=3380,DBID=yyyyy
/*
//DDKARTE DD *
ADARES MERGE CLOG,NUMLOG=3
/*
//
```

VSE/ESA

The following sample job for VSE/ESA systems (see the JOBS data set) illustrates the execution of the ADARES MERGE CLOG function:

```
//JOB ADADESCM
/*
/* ADARES : MERGE SEQUENTIAL COMMAND LOGS
/* FOR USE WITH AN ADABAS NUCLEUS CLUSTER
/*
// LIBDEF PHASE,SEARCH=(SAGLIB.ASM742,SAGLIB.ADA742)
// DLBL ASSOR1,'EXAMPLE.DByyyyy.ASSOR1'
// DLBL DATAR1,'EXAMPLE.DByyyyy.DATAR1'
// DLBL WORKR1,'EXAMPLE.DByyyyy.WORKR1'
// DLBL ASSOR1,'EXAMPLE.DByyyyy.ASSOR1'
// DLBL CLOG01,'EXAMPLE.DByyyyy.CLOGR1.NUC1'
// DLBL CLOG02,'EXAMPLE.DByyyyy.CLOGR1.NUC2'
// DLBL CLOG03,'EXAMPLE.DByyyyy.CLOGR2.NUC3'
// TLBL SIAUS1,'EXAMPLE.DByyyyy.CLOGM'
// ASSGN SYS021,TAPE
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADARES,MODE=MULTI,SVC=svc,DEVICE=3380,DBID=YYYYY
/*
ADARES MERGE CLOG,NUMLOG=3
/*
/&
```

BS2000

The following sample job for BS2000 systems (see the JOBS data set) illustrates the execution of the ADARES MERGE CLOG function:

```

/BEGIN-PROC C
/MOD-TEST DUMP=YES
/REMARK *
/REMARK * A D A R E S MERGE SEQUENTIAL COMMAND LOGS
/REMARK * FOR USE WITH PARALLEL DATABASE
/DEL-FI ADAddd.AUS
/SET-JOB-STEP
/CREATE-FILE ADAddd.AUS,SUP=TAPE(VOL=RES101,DEV-TYPE=T-C4),-
/ PROT=(USER-ACCESS=ALL-USERS)
/SET-JOB-STEP
/ASS-SYSLST L.RES.MERG
/ASS-SYSDTA *SYSCMD
/SET-FILE-LINK DDLIB,ADABAS.MOD
/SET-FILE-LINK DDASSOR1,ADAddd.ASSO ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDDATAR1,ADAddd.DATA ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDWORKR1,ADAddd.WORK ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDCLOG01,ADAddd.CLOG1.NUC1 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDCLOG02,ADAddd.CLOG1.NUC2 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDCLOG03,ADAddd.CLOG2.NUC3 ,SUP=DISK(SHARE-UPD=YES)
/SET-FILE-LINK DDSIAUS1,ADAddd.CLOGM,ACC-METHOD=SAM,-
/ BUFF-LEN=32768,REC-FORM=V,SUP=TAPE(LABEL=STD)
/START-PROG *M(E=ADARUN,L=ADABAS.MOD),RUN-MODE=ADV(A-L=YES)
ADARUN PROG=ADARES,DB=ddd
ADARES MERGE CLOG,NUMLOG=3
/ASS-SYSDTA *PRIM
/ASS-SYSLST *PRIM
/END-PROC

```

ADARES BACKOUT and REGENERATE— Uniquely Identifying Checkpoints

The protection log merge process usually changes the numbering of the PLOG blocks. The PLOG block number of a checkpoint on the original PLOG data set will not necessarily be the same as the block number after the merge. To uniquely identify the checkpoint in this situation, it is necessary to also specify the NUCID for all ADARES functions that can specify a TOBLK / FROMBLK parameter; that is, BACKOUT and REGENERATE.

The merge process ensures that there is at most one checkpoint per block. It records the (old) block number prior to the merge and the NUCID that wrote the checkpoint. When you then specify the block number and NUCID as reported in ADAREP, ADARES is able to uniquely identify the block.



Note: In an Adabas nucleus cluster environment, ADAREP includes the NUCID when printing all checkpoint information.

The additional parameters that are required in an Adabas nucleus cluster environment are `NUCID`, `TONUCID`, and `FROMNUCID`.

If the `NUCID` is the same for the starting and ending checkpoint, only the `NUCID` needs to be specified.



Notes:

1. An `ADAREP CPEXLIST` function can be used to determine the original block number and `NUCID` that wrote the checkpoint. This is the block number prior to the merge and the one that `ADARES REGENERATE` and `BACKOUT` expects.
2. `BACKOUT DPLOG` and `BACKOUT MPLOG` are not allowed for a cluster database. The `PLOG` must be merged before the backout can be performed.

ADASAV Processing Change

Sample JCL is located in the `ADASAVRW` member of the `JOBS` data set.

For the following `ADASAV` functions, the Work data sets/files of all cluster (or noncluster) nuclei for the database that may have been active at the time of the `ABEND` must be reset:

- `RESTONL` (database)
- `RESTONL GCB`
- `RESTORE` (database)
- `RESTORE GCB`

This can be done either:

- manually (e.g., by using `ADAFRM WORKRESET FROMRABN=1,SIZE=1B`); or
- by specifying the Work data sets/files with DD names/link names `DD/WORKRn` (`n=1-9`) or `DD/WORKnn` (`nn=10-32`) in the JCL for the `RESTONL/RESTORE` function.

Otherwise, the nuclei that did not have their Work data sets/files reset will give parm-error 42 when started.

The `DD/PLOGRn` and `DD/CLOGRn` data sets are not reset in the restore process. They must be either copied/merged by `ADARES PLCOPY/CLCOPY` or reset by `ADAFRM`.

ADASAV RESTPLOG -- Uniquely Identifying Checkpoints

After the protection log merge process, the block number will not necessarily be the same. To uniquely identify the checkpoint in this situation, it is necessary to also specify the NUCID parameter for the ADASAV RESTPLOG function when specifying the SYN1 or SYN4 parameter.



Note: An ADAREP CPEXLIST function can be used to determine the original block number and NUCID that wrote the checkpoint. This is the block number prior to the merge and the one that ADASAV RESTPLOG expects.

7

Cluster Nucleus Session End Statistics

▪ General Nucleus Information	40
▪ Input/Output Statistics	40
▪ Command Statistics	41
▪ User Statistics	43
▪ Efficiency Statistics	43
▪ Global Cache Statistics (Cluster Nucleus Only)	44
▪ Global Lock Statistics (Cluster Nucleus Only)	49
▪ Data Set Activity Statistics	55

In addition to the end-of-session statistics printed by every Adabas nucleus, the statistics for a cluster nucleus also include global cache and lock statistics.

If you are running the selectable unit Adabas Online System (AOS), all of the statistics shown in the following sample output are displayed.

If you are running only the demo version of AOS delivered with Adabas, the statistics displayed are limited as follows:

Section	Displays statistics only for...
Global Cache Statistics	totals, DS, and NI
Global Lock Statistics	buffer flush, hold ISN, new data RABN, and global update command sync locks

This chapter covers the following topics:

General Nucleus Information

```
The  A d a b a s  nucleus session
Started  2001-02-13 22:58  and  ended  2001-02-13 23:05

Duration      00000:06:59  hours
Wait-time    00000:02:26  hours
Cpu-time     00000:00:53  hours
```

Input/Output Statistics

- [I/O Counts \(Including Initialization\)](#)
- [Log Reads and Buffer Efficiency](#)
- [Distribution of ASSO/DATA I/Os by Volser Number \(Excluding Initialization\)](#)

I/O Counts (Including Initialization)

READS	WRITES	

ASSO	4710	6913
DATA	1750	2853
WORK	3	7251
PLOG	0	0
CLOG	0	0

Total	6463	17017

Log Reads and Buffer Efficiency

Log. reads	173,393
Buffer eff.	26.8

Distribution of ASSO/DATA I/Os by Volser Number (Excluding Initialization)

Vol-ser	High RABN	Count
WRKM01 (ASSO	8082)	11599
WRKM01 (DATA	5990)	4603
TOTAL		16202

Command Statistics

- [Count of Calls Executed and Threads Used](#)
- [Distribution of Commands by Source](#)
- [Distribution of Commands by Thread](#)
- [Distribution of Commands by File](#)
- [Distribution of Commands by Type](#)

Count of Calls Executed and Threads Used

A d a b a s	executed	10,249 calls
	in	8 threads

Distribution of Commands by Source

Source	Number
Remote commands	0
Local commands	10,102
Internal commands	144
Operator commands	3

Distribution of Commands by Thread

Thread	Number
1	2,657
2	1,803
3	1,401
4	1,300
5	1,193
6	977
7	917
8	1

Total	10,249

Distribution of Commands by File

File	Number
0	4,281
30	5,968

Total	10,249

Distribution of Commands by Type

Cmd-type	Number
A1/4	1,968
CL	44
ET	4,040
N1/2	2,000
OP	43
UC	7
REST	2,147

Total	10,249

User Statistics

```

There were          43 users participating
Most calls (        303) initiated by user  USADFMB2
Most I/O-s (        331) initiated by user  USADFMB2
Most thr.-time (00:00:08) was used by user  USADFMB1

```

Efficiency Statistics

```

46 Formats had to be translated
      0 Formats had to be overwritten
      0 Autorestarts were done
      0 Throw-backs due to ISN problem
      0 Throw-backs due to space problem
     143 Bufferflushes were done

```

- [Buffer Flush Information](#)
- [Actual High-water Marks for Major Pools \(Except the Bufferpool\)](#)

Buffer Flush Information

```

Flush phases                212
Blocks flushed              28,503
Flush I/Os

Flush requests:
Return immediately          52,658
Return after logical flush      0
Return after entire flush      15

```

Actual High-water Marks for Major Pools (Except the Bufferpool)

AREA	ADARUN	PARM	HIGH-WATER-MARK
AB -POOL	NAB=	2000	51712 (0 %)
CQ -POOL	NC =	96000	3840 (4 %)
DUQ -POOL	LDE=	5000	0 (0 %)
FI -POOL	LFP=	20000	6560 (32 %)
HQ -POOL	NH =	16856	588 (3 %)
SC -POOL	LCP=	10000	0 (0 %)
TBI -POOL	LI =	10000	0 (0 %)
TBS -POOL	LQ =	100000	0 (0 %)
UQ -POOL	NU =	500	8844 (6 %)
UQF -POOL	NU =	500	1512 (3 %)
WORK-POOL	LWP=	800000	114296 (14 %)
XID -POOL	XID=		

Global Cache Statistics (Cluster Nucleus Only)

Cast-out dir :	188
Synchronous :	188
Asynchronous :	0
Unlock cast-out:	212
Synchronous :	132
Asynchronous :	80
Directory reads:	3
Synchronous :	0
Asynchronous :	3

- Totals
- Address Converter (AC)
- Data Storage (DS)
- Data Storage Space Table (DSST)
- File Control Block (FCB)
- Normal Index (NI)
- Upper Index (UI)

- File Statistics for Files with More than 25% of the Total Cache Statistics

Totals

Reads	:	15,006
Synchronous	:	15,006
Asynchronous	:	0
In cache	:	6,245
Not in cache	:	8,761
Area full	:	0
Writes	:	66,726
Synchronous	:	66,726
Asynchronous	:	0
Written	:	66,726
Not written	:	0
Area full	:	0
Validates	:	327,623
Block invalid	:	0
Cast-out reads	:	28,503
Synchronous	:	28,503
Asynchronous	:	0
Deletes	:	0
Timeouts	:	0

Address Converter (AC)

Reads	:	8
Synchronous	:	8
Asynchronous	:	0
In cache	:	0
Not in cache	:	8
Area full	:	0
Writes	:	2,004
Synchronous	:	2,004
Asynchronous	:	0
Written	:	2,004
Not written	:	0
Area full	:	0
Validates	:	5,983
Block invalid	:	0

Cluster Nucleus Session End Statistics

Cast-out reads :	72
Synchronous :	72
Asynchronous :	0
Deletes :	0
Timeouts :	0

Data Storage (DS)

Reads :	2,775
Synchronous :	2,775
Asynchronous :	0
In cache :	26
Not in cache :	2,749
Area full :	0
Writes :	4,972
Synchronous :	4,972
Asynchronous :	0
Written :	4,972
Not written :	0
Area full :	0
Validates :	9,965
Block invalid :	0
Cast-out reads :	2,921
Synchronous :	2,921
Asynchronous :	0
Deletes :	0
Timeouts :	0

Data Storage Space Table (DSST)

Reads :	2
Synchronous :	2
Asynchronous :	0
In cache :	0
Not in cache :	2
Area full :	0
Writes :	2,004
Synchronous :	2,004
Asynchronous :	0

Written	:	2,004
Not written	:	0
Area full	:	0
Validates	:	4,490
Block invalid	:	0
Cast-out reads	:	69
Synchronous	:	69
Asynchronous	:	0
Deletes	:	0
Timeouts	:	0

File Control Block (FCB)

Reads	:	5
Synchronous	:	5
Asynchronous	:	0
In cache	:	0
Not in cache	:	5
Area full	:	0
Writes	:	4,970
Synchronous	:	4,970
Asynchronous	:	0
Written	:	4,970
Not written	:	0
Area full	:	0
Validates	:	56,029
Block invalid	:	0
Cast-out reads	:	119
Synchronous	:	119
Asynchronous	:	0
Deletes	:	0
Timeouts	:	0

Normal Index (NI)

Reads	:	12,057
Synchronous	:	12,057
Asynchronous	:	0
In cache	:	6,219
Not in cache	:	5,838
Area full	:	0
Writes	:	44,096
Synchronous	:	44,096
Asynchronous	:	0
Written	:	44,096
Not written	:	0
Area full	:	0
Validates	:	25,685
Block invalid	:	0
Cast-out reads	:	22,973
Synchronous	:	22,973
Asynchronous	:	0
Deletes	:	0
Timeouts	:	0

Upper Index (UI)

Reads	:	159
Synchronous	:	159
Asynchronous	:	0
In cache	:	0
Not in cache	:	159
Area full	:	0
Writes	:	8,680
Synchronous	:	8,680
Asynchronous	:	0
Written	:	8,680
Not written	:	0
Area full	:	0
Validates	:	225,471
Block invalid	:	0
Cast-out reads	:	2,349

Synchronous	:	2,349
Asynchronous	:	0
Deletes	:	0
Timeouts	:	0

File Statistics for Files with More than 25% of the Total Cache Statistics

File	30:	
Reads	:	14,998
Writes	:	64,710
Validates	:	323,105

Global Lock Statistics (Cluster Nucleus Only)

- General Control Block (GCB) Lock
- Security Lock
- File Space Table (FST) Lock
- File Lock Table Lock
- Online Save Lock
- Buffer Flush Lock
- Global ET Sync Lock
- Recovery Lock
- Hold ISN Locks
- Unique Descriptor Locks
- ETID Locks
- New Data RABN Locks
- Checkpoint Lock
- ET Data Lock
- Global Update Command Sync Lock
- Parameter Lock

General Control Block (GCB) Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0

Cluster Nucleus Session End Statistics

Synchronous	:	0
Asynchronous	:	0

Security Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

File Space Table (FST) Lock

Obtains - Conditional	:	1
Granted	:	1
Rejected	:	0
Unconditional	:	1
Synchronous	:	2
Asynchronous	:	0
Releases - Issued	:	2
Synchronous	:	2
Asynchronous	:	0

File Lock Table Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	5
Synchronous	:	5
Asynchronous	:	0
Releases - Issued	:	5
Synchronous	:	5
Asynchronous	:	0

Online Save Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

Buffer Flush Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	152
Synchronous	:	152
Asynchronous	:	0
Releases - Issued	:	152
Synchronous	:	152
Asynchronous	:	0

Global ET Sync Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

Recovery Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

Hold ISN Locks

Obtains - Conditional	:	3972
Granted	:	3972
Rejected	:	0
Unconditional	:	0
Synchronous	:	3972
Asynchronous	:	0
Releases - Issued	:	3972
Synchronous	:	3972
Asynchronous	:	0

Unique Descriptor Locks

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

ETID Locks

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

New Data RABN Locks

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	1000
Synchronous	:	1000
Asynchronous	:	0
Releases - Issued	:	1000
Synchronous	:	1000
Asynchronous	:	0

Checkpoint Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	4
Synchronous	:	4
Asynchronous	:	0
Releases - Issued	:	4
Synchronous	:	4
Asynchronous	:	0

ET Data Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

Global Update Command Sync Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	143
Synchronous	:	143
Asynchronous	:	0
Releases - Issued	:	143
Synchronous	:	143
Asynchronous	:	0

Parameter Lock

Obtains - Conditional	:	0
Granted	:	0
Rejected	:	0
Unconditional	:	0
Synchronous	:	0
Asynchronous	:	0
Releases - Issued	:	0
Synchronous	:	0
Asynchronous	:	0

Data Set Activity Statistics

```
.....  
ADAI03 DDWORKR1      3 READS      7251 WRITES  
ADAI03 DDDATAR1    1750 READS    2853 WRITES  
ADAI03 DDASSOR1    4710 READS    6913 WRITES
```


8 Performance and Tuning

- ADARUN Parameter Settings 58
- Allocating Work Data Set Space 58
- Using Close (CL) Commands 58
- Deferred Cache Operations 59
- Tuning Buffer Flushes 60
- Optimizing Global Cache and Lock Areas 60
- Minimizing Communication with Global Areas 62
- Optimizing Block Sizes 63

This chapter covers the following topics:

ADARUN Parameter Settings

Software AG recommends that you use your existing Adabas ADARUN parameters (or the default values) for each nucleus in an Adabas cluster, and then tune the values after analyzing the performance of the cluster.

Session statistics can be used to determine the best settings for each parameter. The statistics can be displayed using operator commands during the session; the statistics are also printed automatically at the end of a session or in response to an ADADBS REFRESHSTATS command.

For parameters that allot processing resources to the cluster nuclei (such as NU, NH, LP, etc.), Software AG recommends that you set them large enough that each individual cluster nucleus could handle the entire load on the database if the other nuclei were to terminate abnormally.

Allocating Work Data Set Space

Each Adabas cluster nucleus requires its own Work data set to hold its temporary data.

The individual sizes of the different Work parts (1, 2, and 3) as specified by ADARUN parameters such as LP and LWKP2 can be different among the nuclei; however, the overall size of each Work data set must be the same, because the total Work size is stored in the Adabas general control block (GCB). Software AG recommends that you use the same LP and LWKP2 values on each nucleus active for the same database.

For each nucleus, you need to specify shared access to DD/WORKR1. During an offline or online recovery, a nucleus may access the Work data sets belonging to other nuclei in the cluster.

Using Close (CL) Commands

Users are assigned to a nucleus for their entire sessions and should therefore issue Adabas close (CL) commands as appropriate. The close command ends the user's session, making the user eligible for reassignment to another nucleus when the user again issues an Adabas open (OP) command. This allows Adabas Parallel Services to rebalance the workload over the participating nuclei.

Deferred Cache Operations

Publication of updated blocks to the global cache area is usually deferred until just before the end of the associated transaction. Multiple updates to a block may produce only a single write of the block to the cache rather than a cache write for each update.

The greater the number of database update in parallel transactions, the greater the expected improvement in performance.



Note: Deferred cache operations create an asymmetry between users on the update nucleus, who see uncommitted updates (unless they read with hold), and users on other cluster nuclei, who may or may not see uncommitted updates.

- [Redo Pool](#)
- [ADARUN Parameter LRDP](#)

Redo Pool

Since the write of updated blocks to the cache may fail due to conflicting updates to the same blocks by other nuclei in the cluster, every cluster nucleus must be capable of redoing the updates it has not yet written to the cache. The nucleus maintains information about these updates in the "redo pool".

ADARUN Parameter LRDP

The size of the redo pool is specified by the new ADARUN parameter LRDP. The LRDP parameter is effective only in a cluster nucleus, that is, when a nonzero NUCID is specified.

If LRDP is not specified, the nucleus takes as default the value of the LFIOP parameter. If LRDP is explicitly set to zero, the nucleus writes each update immediately to the cache.

Different nuclei in the same cluster can have different settings of LRDP. It is also possible, although not recommended, to run one nucleus with LRDP=0 and a peer nucleus with LRDP>0.



Note: If one nucleus runs with LRDP=0 and a peer nucleus runs with LRDP>0 and the different cluster nuclei concurrently update the same Data Storage blocks, incorrect DSST entries may be produced. These are reported by ADADCK. Such errors are harmless and do not affect the results of the application programs.

The nucleus reports on the use (high watermark) of the redo pool in a shutdown statistic and in the response to the DRES command from the operator console or from ADADBS OPERCOM.

Tuning Buffer Flushes

When the update load on the database is so high that the buffer flush becomes the bottleneck, you can improve performance by reducing the duration of buffer flushes.

Instead of starting one I/O per volume, a buffer flush can initially start a predetermined number of I/Os on each volume and then start a new one when another I/O on the same volume finishes. This occurs independently on each volume.

The ADARUN parameters `LFIOP` and `FMXIO` (see the *Adabas Operations* documentation for details) can be used to control buffer flushes. The `LFIOP` parameter enables asynchronous buffer flush operation and sets the I/O pool size. The `FMXIO` parameter sets the limit on the number of I/O operations that can be started in parallel by `LFIOP` flush processing.

- [Effect of ASYTVS Parameter Setting](#)
- [Dynamically Modifying the FMXIO Parameter Setting](#)

Effect of ASYTVS Parameter Setting

The meaning of the `FMXIO` parameter is affected by the setting of the `ASYTVS` parameter:

When `ASYTVS=YES` (buffer flushes occur by volume), `FMXIO` specifies the number of I/Os to be started in parallel *on each volume*. The minimum and default number is 1; the maximum number is 16. If you specify a number greater than 16, it is reduced to 16 without returning a message.

When `ASYTVS=NO` (buffer flushes occur in ascending RABN sequence without regard to the distribution of the blocks over volumes), the minimum, default, and maximum values continue to be 1, 60, and 100, respectively.

Dynamically Modifying the FMXIO Parameter Setting

The setting of `FMXIO` can be modified dynamically using the `FMXIO=nn` command from the operator console or the Modify Parameter function of Adabas Online System.

Optimizing Global Cache and Lock Areas

As a user, you must allocate and define sizes that are appropriate to your application needs for the global cache and lock areas.

This section provides guidelines for determining optimal sizes for these areas based on current experience.



Note: There may be sites for which these guidelines are not appropriate.

- [Global Cache Area Size](#)
- [Global Lock Area Size](#)

Global Cache Area Size

The global cache area must be large enough to retain:

- *directory elements* for all blocks that reside in all the buffer pools; and
- enough data elements to keep the changed blocks between buffer flushes (cast-outs).

Directory elements are used to keep track of the cluster members that have a particular block in their buffer pools so that the block can be invalidated should any member modify it.

If the number of directory elements is insufficient, Adabas Parallel Services reuses existing directory elements and invalidates the blocks associated with those directory elements, because they can no longer be tracked. These blocks must then be reread from the database and registered again the next time they are referenced and validated, even though they did not change.

It is generally better to reassign storage for data elements to keep more ASSO and DATA blocks in the global cache area than to define too many directory elements in the global cache area. More data elements than necessary can be used to keep additional blocks to improve the local buffer efficiency.

The number of directory elements need not be greater than the sum of the sizes of all buffer pools divided by the smallest block size in use for ASSO and DATA.

When connecting to the global cache area during startup, the ADAX57 message reports the number of directory elements and data elements. The ADARUN parameters `DIRRATIO` and `ELEMENTRATIO` determine the ratio between the number of directory and data elements.

Global Lock Area Size

All nuclei in a database cluster share the global lock area.

Locks are held for a variety of entities, for example unique descriptor values. These lock types tend to occur with very different frequencies. The amount of lock activity during a session for each lock type is displayed in the shutdown statistics.

It is often the case that ISN locks show the greatest activity. The sum of high-water marks for NH yields an upper limit for the number of ISN locks that were held concurrently during the session.

The global lock manager uses a hash table to allocate and find a specific lock entry.

When the global lock manager receives a lock request (for example, to put an ISN of a file into hold status), it allocates a specific lock entry unless another member of the cluster has already

made a conflicting allocation. A conflicting allocation produces lock contention because another member holds the same lock. Depending on its type, the lock request is then rejected or remains pending, waiting for the associated resource to become available.

The minimum lock structure size can be roughly estimated as:

```
(NU*3 + NH + LDEUQP/16 + MAXFILES*4 + 50) * 240 + 500,000 bytes
```

where `MAXFILES` is the maximum number of files in the database (set in `ADADEF` or `ADAORD`) and `NU`, `NH`, and `LDEUQP` are the `ADARUN` parameters of the cluster nuclei. The formula in parentheses $(NU*3 + NH + LDEUQP/16 + MAXFILES*4 + 50)$ is used to calculate the minimum number of lock records that the cluster nuclei expect to have available.

Minimizing Communication with Global Areas

Most of the additional processing required for Adabas Parallel Services environments compared to a single Adabas nucleus involves communication with the global areas.

For this reason, optimizing the performance of an Adabas Parallel Services environment means minimizing the need for communication with the global areas. It is also important to keep the time required for each communication as short as possible.

- [Avoiding the Hold Option](#)
- [Reducing Direct Interaction with the Global Cache Area](#)

Avoiding the Hold Option

Lock requests usually depend on application requirements. Under data-sharing, the hold option is more expensive and access with the hold option should be avoided unless records will in fact be updated or must be protected from concurrent updates.

Reducing Direct Interaction with the Global Cache Area

Cache area requests occur when blocks:

- that are referenced do not exist in the local buffer pool;
- exist in the local buffer pool but have become invalid due to concurrent updates by other cluster members or from directory reuse; or
- are updated.

The first and second situation require registering and (re)reading the blocks from the global cache area. This is more expensive than just validating blocks.

The first situation is related to the buffer efficiency in a noncluster environment. In a cluster environment, buffer efficiency represents the combined effect of the local buffer pool and the global cache area. In order to reduce the interaction with the global cache, the local buffer pool (LBP) should not be decreased from what would be used in a noncluster nucleus. A large LBP parameter and the usage of forward index compression are recommended to improve the buffer efficiency in the local buffer pool.

Optimizing Block Sizes

Although earlier versions of Adabas often worked well with large block sizes, the buffer pool manager and forward index compression features introduced with Adabas version 7 make smaller block sizes more attractive, especially in data-sharing mode.

Use the following guidelines when selecting an optimal block size for ASSO and DATA:



Note: Only general recommendations can be given.

1. Avoid 4-byte RABNs

If the database is not extremely large, avoid 4-byte RABNs as this increases the number of AC blocks by 33%. When growth considerations are taken into account, this may require larger block sizes or limit reductions in block size. The same holds true for the maximum compressed record length.

2. Use forward index compression

Forward index compression can significantly reduce the number of index blocks in a database. Apply forward index compression to all frequently accessed files (or to all files, regardless of their frequency of use). Choose the ASSO block size that is as small as possible but large enough to keep the number of index levels down to 3 or 4.

3. Minimize frequently updated descriptors

When files are updated frequently, the number of blocks that are modified and need to be written to the global cache area often depends on the number of descriptors that have been defined and modified during update processing. Support for additional keys whose descriptor values are subject to frequent modifications becomes even more expensive in a data-sharing environment.

Index

A

- ADACOM
 - starting, 4
- ADARES utility
 - MERGE CLOG function, 33
- ADARUN parameters
 - determining correct settings, 58

C

- cache and lock size, estimating, 62
- CLnn command
 - close a nucleus to new users, 58
- COMPRINT
 - messages written to, 4

N

- NUCID
 - ADADBS utility OPERCOM parameter
 - for command routing, 21
- Nucleus
 - routing utility-issued operator commands, 21
 - serialization during initialization, 5
 - starting, 4

P

- Parameters
 - ADADBS OPERCOM
 - NUCID, 21

S

- Serialization
 - of nucleus, 5

U

- Utilities
 - ADADBS OPERCOM
 - NUCID parameter, 21
 - cluster environment, 20

