Adabas Parallel Services

Concepts and Facilities

Version 8.6.1

October 2025

ADABAS & NATURAL

This document applies to Adabas Parallel Services Version 8.6.1 and all subsequent releases.

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

Document ID: ASM-INTRO-861-20251030

Table of Contents

Preface

Adabas Parallel Services is a selectable unit of Adabas that helps you make full use of multiple-engine processors by allowing a single physical database to be accessed simultaneously by up to 31 nuclei.

This document is organized as follows:

What is Adabas Parallel Services?	Provides a high-level description of the product.
Adabas Parallel Services and Adabas Cluster Services	Describes the differences between Adabas Parallel Services and Adabas Cluster Services.
Assigning Users	Describes how Adabas Parallel Services assigns users to nuclei in a cluster.
Cluster Tasks	Describes the types of tasks that comprise an Adabas Parallel Services cluster.
Global Areas	Describes global area usage in Adabas Parallel Services.
Global Messaging Services	Describes the messaging services that are essential to communication among the active nuclei of an Adabas Parallel Services cluster.
Global Commands	Describes the global operator commands available with Adabas Parallel Services.
Logs and Log Merges	Describes the protection log (PLOG) and command log (CLOG) usage in Adabas Parallel Services.
Parallel Participant Table (PPT)	Discusses the use of the Parallel Participant Table (PPT) in Adabas Parallel Services.
Dynamic Allocation	Describes the use of dynamic allocation by ADARES for Adabas Parallel Services data sets and files.
Components Summary	Provides a component summary for Adabas Parallel Services.
Switching Between Cluster and Noncluster Modes / PLOG Handling	Describes switching between cluster and non-cluster modes in Adabas Parallel Services.
Adabas Parallel Services and Other Software GmbH products	Describes the support for Adabas Parallel Services provided by other Software GmbH products.
Restricted Support for Adabas Features	Describes facilities that are not supported for cluster nuclei running under Adabas Parallel Services or for which restricted support is provided.
Getting Started	Describes how to get started using Adabas Parallel Services and where to go for more information.

1 Conventions

This section covers the following topics:

Syntax

This following syntax conventions are used in this documentation when describing ADARUN parameters and operator commands:

- The vertical bar (|) separates parameter options or values, one of which must be chosen.
- Curly brackets or braces ({ }) delimit the series of options or values.
- (Square) brackets ([ ]) indicate a term that may optionally be specified.
- The underline (_) identifies the default value.
- Three dots preceded by a comma or a slash (, ... or / ...) indicates that the previous term can be repeated but must be preceded by the comma or slash as indicated.
- A range of values is indicated by two instances of a variable (for example, "fnr", "rabn", "csn") separated by a hyphen (for example, "fnr - fnr"). The first instance represents the low value and the second the high value of the range. The range specification comprises the two numbers separated by the hyphen with no intervening spaces.
- Parentheses (( )) are part of the term and must be included as part of the parameter statement.

2 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

3 What is Adabas Parallel Services?

Adabas Parallel Services implements multinucleus, multithread parallel processing and optimizes Adabas in a multiple-engine processor environment on a single operating system image.

Up to 31 Adabas nuclei in an Adabas Parallel Services cluster are distributed over the multiple engines provided by the system.

All nuclei in the cluster access a single physical database simultaneously. A "single physical database" is one set of Associator and Data Storage data sets identified by a single database ID number (DBID). Each nucleus writes its protection data to its own private work data set and protection logs (PLOGs). Each nucleus also records the commands it executes in its own command logs (CLOGs).

The nuclei communicate and cooperate with each other to process the users' work. Compression, decompression, format buffer translation, sorting, retrieving, searching, and updating operations can all occur in parallel.

In addition to the increased throughput that results from parallel processing, Adabas Parallel Services increases database availability during planned or unplanned outages: the database can remain available when a particular cluster nucleus requires maintenance or goes down unexpectedly.

A practically unlimited number of Adabas Parallel Services clusters can operate in the same operating system image under the same or different SVCs or IDTNAMEs; that is, a practically unlimited number of separate databases can be processed, each with its own Adabas Parallel Services cluster of up to 31 nuclei.

Applications see only one database target; no interface changes are required. Applications still communicate with their intended databases and communicate with an Adabas Parallel Services cluster of nuclei without modification.

4 Adabas Parallel Services and Adabas Cluster Services

Adabas Parallel Services is very similar to another Adabas add-on product, Adabas Cluster Services. As shown in the following table, the differences between the two are the mechanisms used for inter-nucleus communication and for maintaining and accessing the shared cache and lock data:

Service	Adabas Parallel Services	Adabas Cluster Services
Architecture	all nuclei on same system image	nuclei on different system images in parallel sysplex
Communication (inter-nucleus)	via inter-process communication (ADALNK)	via cross-system coupling (XCF) services
Cache	in shared data space or 64-bit addressable virtual storage	in coupling facility
Lock	in shared data space	in coupling facility

5 Assigning Users

When a user program issues the first command to the cluster database, the user is assigned to an active nucleus in the cluster. All commands from that user are routed to the assigned nucleus until the user issues a close (CL) command. At this point, the user is free to be assigned to another nucleus with the next open (OP) command.

Adabas Parallel Services keeps track of the number of users assigned to each nucleus and assigns new users to nuclei evenly to balance the load.

6 Cluster Tasks

Adabas Parallel Services ensures data integrity and enables intercommunication among the Adabas nuclei in each cluster.

All nuclei in a cluster share the same database (ASSO and DATA data sets, DBID), global cache and lock areas, and global nucleus and user table.

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 Adabas Parallel Services characteristics of a particular DBID are fully transparent to the Adabas link routines.

Two types of tasks together build an Adabas Parallel Services cluster:

- a control task (ADACOM);
- one or more nuclei up to a maximum of 31.

PLCBX Structures

Adabas Parallel Services uses a "PLXCB" and subordinate data areas (referred to as "PLXCB structures") for maintaining information about active users and the nuclei in the cluster to which they are assigned. The PLXCB structures are identified by the DBID of the related cluster database and the Router ID (SVC number) through which users sends Adabas calls to the nuclei of that database. They are allocated by the first ADACOM task to which the Router ID / DBID combination is defined.

By default, the PLXCB structures are allocated in 31-bit common storage (ECSA). Such PLXCB structures stay in existence even after the ADACOM task that allocated them has ended. They exist for the life of the operating system image or until they are explicitly deleted via **ADACOM**.

The PLXCB structures can be sizable, depending on the number of users supported - but still less than 1 MB of storage for every 10,000 users configured (NU parameter) -, and an installation may have multiple Parallel Services databases with each one requiring its own PLXCB structures. Common storage is a finite resource shared among all users in the system and may become constrained.

On z/OS, the PLXCB structures may also be placed in a dataspace owned by ADACOM, by specifying LOC=DSP for the SVC/DBID combination given to ADACOM. Placing PLXCB structures in ADACOM dataspaces provides relief from 31-bit ECSA storage constraints.

Note that ADACOM creates the dataspace for the PLXCB structures of each cluster database as a common storage dataspace (SCOPE=COMMON). The number of common storage dataspaces in a system is limited by the MAXCAD parameter in SYS1.MACLIB(IEASYS*xx*), which can range from 10 through 250 and has a default value of 50. This limits the number of cluster databases on one operating system image that can have their PLXCB structures allocated with LOC=DSP.

ADACOM

Note: Router ID is the SVC number.

The ADACOM task must be run to set up the Adabas Parallel Services cluster environment before any cluster nucleus is started. The PLXCB structures are allocated for maintaining information about active users and the nuclei to which they are assigned.

ADACOM also holds the cache and lock data spaces that the cluster nuclei use for sharing information. The data spaces can exist only as long as their owner, ADACOM, is active. For this reason, ADACOM cannot terminate if it is holding data spaces for an active Adabas Parallel Services cluster. Note that with Adabas Cluster Services, ADACOM does not have to be active all of the time.

Important: Cancelling ADACOM causes all Adabas Parallel Services nuclei it manages to fail.

One ADACOM can manage multiple cluster databases on different routers. ADACOM parameters specify the Router ID / DBID combinations (sets) that the ADACOM is to manage:

- The Router ID identifies the SVC number on z/OS systems. The Router ID value must be the same within a cluster; however, the same Router ID may be used for different clusters.
- The DBID identifies the external physical database shared by a particular cluster of nuclei and known to the application.

You can dynamically add or terminate Router ID / DBID combinations during an Adabas Parallel Services session. ADACOM itself, however, cannot be terminated as long as any of the Router ID / DBID sets it manages is still active.

Although a single ADACOM job can run all Router ID / DBID sets in a cluster environment, it is possible to run multiple ADACOM tasks simultaneously with the same, mixed, or completely different Router ID / DBID sets. Each ADACOM task with the same Router ID / DBID set will attach a subtask for operator commands and other functions. Only the first ADACOM for that set will attach a second subtask to own any data spaces.

A COMPRINT data set is used to hold global messages that apply to all Router ID / DBID sets defined to an ADACOM task.

In addition, Adabas Parallel Services dynamically allocates a spool data set for command output to each Router ID / DBID set. The format of its DD name is `Pssddddd` where `ss` is the last two digits of the SVC number and `ddddd` is the DBID.

Alternatively, for each Router ID / DBID combination, you can set up an output data set in the startup JCL by specifying the DD name in the format `Pssddddd` as just explained. The logical record

length (LRECL) must be set to 80. The command output for the corresponding Router ID / DBID combination, whether defined in the startup JCL or dynamically allocated, is written to that data set. You might want to preset the spool output modules to control the data set's DD name or the message class, which otherwise defaults to "X".

When you dynamically terminate a Router ID / DBID combination, the related spool output module is retained.

The first ADACOM started governs the size of the global user table (`NU` parameter): different values set for subsequent ADACOMs or for nuclei are ignored. If `NU` is set to zero, ADACOM frees the environment. If you do not set a value for NU, it defaults to 200. ADACOM parameters are discussed in the section *Initialization Parameters* in the *Adabas Parallel Services Reference Guide*.

After initialization, ADACOM functions as a command manager to monitor and control cluster nuclei running under Router ID / DBID sets it manages and to dynamically create or terminate Router ID / DBID sets. ADACOM commands are available to:

- exercise some control over the assignment of users to nuclei
- display the number of commands and user assignments for each active nucleus
- dynamically create or terminate Router ID / DBID combinations (sets).

Cluster of Adabas Nuclei

The ADARUN parameter `CLUSTER=LOCAL` is used to identify a nucleus as a member of an Adabas Parallel Services cluster. The `NUCID` parameter identifies the Adabas Parallel Services cluster nucleus for communication via the Adabas router. The `NUCID` must be a unique number in the range 1-65000. A non-cluster nucleus has the parameter `CLUSTER=NO` (the default value); its `NUCID` is zero.

The `NUCID` is the only ID table entry for an Adabas Parallel Services nucleus.

Additionally, each nucleus has its own:

- command queue
- local buffer pool (LBP) and manager;
- Work data set, which must have the same device type and physical size for each nucleus in the cluster;
- dual or multiple protection log (PLOG) data sets (optional)
- dual or multiple command log (CLOG) data sets (optional)
- workpool
- format pools, etc.

Dual or multiple protection logs are optional but if supplied, each nucleus must have its own and all must use the same type. The same is true for command logs.

The nucleus handles all updates and protection logging. Updates are handled in a timely manner to ensure that the other cluster nuclei access the most recent data.

The ADADBS OPERCOM functions provide the `NUCID` parameter to route the command to a specific nucleus in the Adabas Parallel Services cluster. If `NUCID` is not specified, the command goes either to the nucleus assigned to ADADBS or AOS, or to all nuclei, depending on the inherent logic of the command.

Router Cluster Component SVCCLU

The Adabas router, (ADASVC) always contains the SVCCLU component. The SVCCLU component includes functionality for Adabas cluster environments. The resulting SVC routes commands to the cluster nuclei in an Adabas Parallel Services cluster. To make routing decisions, it uses the nucleus and user tables in local common storage, which are updated based on nucleus or ADACOM information.

There is no predefined limit to the number of Adabas Parallel Services clusters, each servicing a separate database, that can run under a single router (ID table).

SVCCLU is invoked by the router as an "exit". It:

- maintains a table of active users and the nucleus to which each is currently assigned;
- assigns new users to the least loaded active cluster nucleus; and
- routes user commands according to the established assignment to cluster nuclei.

7 Global Areas

Adabas Parallel Services uses global areas in dataspaces for cluster caching and locking functions. Under z/OS version 1 release 5 or later releases, cache can also be maintained in shared 64-bit addressable virtual storage.

The global cache area ensures that current data is available to all nuclei in a cluster. It helps keep Associator and Data Storage blocks in the local buffer pools up to date.

The global lock area is used to protect the resources needed during command execution against conflicting use by multiple cluster nuclei.

Global cache and lock areas are sized in each nucleus using the ADARUN parameters `CLUCACHESIZE` and `CLULOCKSIZE`, respectively.

The first cluster nucleus that starts provides its `CLUCACHE`*`xxxx`* and `CLULOCK`*`xxxx`* parameters to ADACOM, which uses a subtask to allocate the global areas for the cluster in which the nucleus participates. In addition, ADACOM dynamically allocates a data set to each cluster for cluster-related message output if the file is not defined in the ADACOM startup JCL.

Note: Read *Performance and Tuning*in the *Adabas Parallel Services Operations Guide* for a formula for estimating cache and lock size.

ADACOM maintains the global areas. It refuses to terminate normally as long as it owns any global areas, because doing so would cause all active nuclei in all clusters using these global areas to fail.

For each Parallel Services cluster, ADACOM prints dataspace-related messages to an output data set/file with the DD name/link name `Dssddddd`, where *`ss`* is the last two digits of the SVC number and *`ddddd`* is the DBID. On z/OS systems, ADACOM automatically allocates this data set in the spool with SYSOUT=X, if it is not explicitly specified.

The global areas are not persistent; that is, they disappear when the last active nucleus terminates, whether normally or abnormally. No manual cleanup is required.

Global Cache Area and Manager

A global cache area is allocated to each Adabas Parallel Services cluster for ASSO and DATA blocks that have been updated during the session.

This section covers the following topics:

- Local Buffer Pool and Manager
- Global Cache and Manager
- Buffer Flush

- Global Cache Storage Options

Local Buffer Pool and Manager

Every nucleus in an Adabas cluster has a local buffer pool and manager.

The buffer pool manager oversees all nucleus requests for reading and writing Associator and Data Storage blocks. For each block in its local buffer pool, the buffer pool manager:

- registers its interest in the block with the global cache manager;
- checks the global cache manager for the status of the registered block to ensure that its nucleus always has the most current copy of the block; and
- writes changed blocks to the global cache area.

The size of the local buffer pool of each nucleus is determined by the ADARUN parameter `LBP`. It must be large enough to hold the active working set of database blocks being used by the nucleus at any one time.

Global Cache and Manager

The global cache area must be large enough to hold the active working set of blocks from the local buffer pool of each nucleus in the cluster. The global cache manager oversees all requests for Associator and Data Storage blocks and copies changed blocks between the local buffer pools and the global cache area to maintain data integrity.

When a nucleus requests a block, it checks the global cache area as well as its local buffer pool to locate the block. If an up-to-date copy of the block is already in the local buffer pool, it is used straight away. Otherwise, if the block is in the global cache area, it is copied to the local buffer pool. If neither is the case, the block is read in from the database. In any case, the global cache manager keeps track of the existence of the block in the local buffer pool and invalidates the local copy if another nucleus updates the same block.

The global cache manager also handles the deletion of blocks in the global cache area when it becomes necessary to reclaim the space they occupy.

Buffer Flush

Any active nucleus may perform buffer flushes.

The buffer flush accommodates the fact that all updated blocks are located in the global cache area. From the global cache area, modified blocks are "cast out" to the flush I/O pool (FIOP) buffer before they are written to disk. The FIOP buffer is sized using the ADARUN `LFIOP` parameter, and the frequency of buffer flushing depends on the limit set. Until such blocks are written to the database, the global cache area holds more current information than the database.

Global Cache Storage Options

Global cache data can be maintained in a dataspace. Support for other storage options for global cache data varies based on the version of z/OS running in your environment:

- On z/OS systems running version 1 release 5 or later, global cache data can be maintained in shared 64-bit addressable virtual storage.
- On z/OS systems running version 1 release 9 or later, global cache data can be maintained in shared 64-bit virtual storage that is backed by page-fixed one-megabyte (1M) large pages.
- On z/OS systems running version 2 release 1 or later, global cache data can be maintained in shared 64-bit virtual storage that is backed by page-fixed two-gigabyte (2G) large pages .

When a dataspace is used for global cache, the maximum size is 2G, a limit imposed by the operating system. Using shared 64-bit addressable virtual storage removes this virtual storage constraint and extends the maximum size of the cache from 2G to tens of GB.

Note: Virtual 64-bit storage backed by 1M or 2G large pages can only be used on IBM systems for which IBM large page support has been enabled and provided the large page pool has been configured to a sufficient size and is available in the system. You can allocate the size of the large page pool using the LFAREA parameter in the IEASYS*xx* member of SYS1.PARMLIB.

Use the ADARUN `CLUCACHETYPE` parameter to specify the virtual storage type for the global cache. The default is "DSP", indicating a dataspace of the size specified by the ADARUN `CLUCACHESIZE` parameter will be used for both control structures and cached data.

The other values (valid only on z/OS systems) for the `CLUCACHETYPE` parameter are:

- "V64" (on z/OS v1.5 or later versions), which indicates that the `CLUCACHESIZE` parameter should specify the amount of shared 64-bit virtual storage that will be used for both control structures and cached data.
- "L64" (on z/OS v1.9 or later versions), which indicates that the `CLUCACHESIZE` parameter should specify the amount of shared 64-bit virtual storage, backed by page-fixed one-megabyte (1M) large pages, that will be used for both control structures and cached data. If unsufficent large pages are available, the shared 64-bit virtual storage will be backed by pageable four-kilobyte (4K) pages.
- "G64" (on z/OS v2.1 or later versions), which indicates that the `CLUCACHESIZE` parameter should specify the amount of shared 64-bit virtual storage, backed by page-fixed two-gigabyte (2G) large pages, that will be used for both control structures and cached data. If unsufficent large pages are available, the shared 64-bit virtual storage will be backed by pageable 4-kilobyte (4K) pages.

To use the 64-bit global cache, your systems programmer must enable shared 64-bit virtual storage in SYS1.PARMLIB. To use large pages, your systems programmer must enable large pages in SYS1.PARMLIB.

Global Lock Area and Manager

Each Adabas Parallel Services cluster uses a global lock area to manage the setting, status, and release of various locks imposed during multiple update nucleus processing. The global lock manager synchronizes the nuclei, users, and transaction processing to ensure data integrity.

The global lock manager is used to:

- connect to and disconnect from the global lock area;
- obtain (conditional or unconditional), release, and alter the ownership (shared or exclusive) of resource locks; and
- read recovery information about a failed peer nucleus.

Lock manager calls may be asynchronous, meaning that the nucleus may continue processing in other threads before a call has completed.

If a lock request is conditional, it is rejected if the lock is not free; if the lock request is unconditional, the nucleus thread waits until the requested lock is free before continuing.

In general, locks are used to prevent two cluster nuclei from using the same resource at the same time. Such resources include:

- data records, which are protected by hold queue element (HQE) locks;
- unique descriptor values, which are protected by unique descriptor element (UQDE) locks;
- end-transaction IDs, which are protected by the ETID lock; and
- various other single-instance resources.

8 Global Messaging Services

Messaging services are essential to communication among the active nuclei of an Adabas Parallel Services cluster. Parallel Services nuclei communicate among one another using the standard Adabas communication environment.

For example, messages need to be sent between nuclei when:

- the general control block (GCB) is modified
- a global parameter is modified
- a buffer flush occurs
- a PLOG or CLOG switches.

Adabas Parallel Services messaging statistics are produced during normal nucleus termination and reported using the ADAX14 and ADAX15 messages. For more information, read your *Adabas Messages and Codes Manual*.

9 Global Commands

For routing operator commands to all nuclei in an Adabas Parallel Services cluster, the following console operator and ADADBS OPERCOM commands have a "GLOBAL" option:

ADAEND
CANCEL
FEOFCL
FEOFPL
HALT

When "GLOBAL" is specified, the nucleus receiving the command passes it on to all active nuclei in the cluster. When "GLOBAL" is not specified for a command from this list, the command applies only to the nucleus receiving it. This is either the nucleus directly addressed by the command (console operator command or ADADBS or Adabas Online System command with NUCID parameter) of the nucleus to which ADADBS or AOS was assigned by Adabas Parallel Services.

10 Logs and Log Merges

In an Adabas cluster environment, the protection logs (and optionally, the command logs) of all individual nuclei in the cluster are 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 private protection and command records. Once the private records are merged, the timestamps are removed. The timestamp is the only difference between an unmerged CLOG (that is, private) and merged CLOG record. The PLOG datasets are the same between cluster and non-cluster.

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 NUCID of the nucleus that writes the checkpoint.

ADARES is the only utility that handles both the unmerged and merged PLOGs and CLOGs. The log merge process:

- merges PLOG or CLOG records into correct chronological sequence.
- avoids a single point of failure by partially merging as much PLOG or CLOG data as possible if a nucleus is unable to respond to a switch.
- is automatic; merges directly from the dual or multiple PLOGs or CLOGs, which eliminates the need to retain control information for tracking unmerged logs.
- quickly copies and merges so that the PLOG or CLOG can be reused.
- uses an identical PLOG or CLOG record layout for an unmerged or a merged record.
- produces a consistent amount of data from each PLCOPY or CLCOPY.

For more detailed information, read *Adabas Parallel Services Operations* in the *Adabas Parallel Services Operations Manual* and read about the ADARES utility in the *Adabas Utilities Manual*.

Protection Logs

Either all nuclei in an Adabas Parallel Services cluster must run without protection logs (PLOGs) or all nuclei must run with PLOGs. The same ADARUN PLOGRQ setting is enforced for all nuclei in a cluster. This value cannot be modified during a session.

If the nuclei in the cluster are running with PLOGs, they must be dual or multiple PLOGs: sequential PLOGs are not allowed. For more information, read your *Adabas Operations Manual*.

Each nucleus in the cluster writes its protection data to its own Work and PLOG data sets: no synchronization is required. However, every protection record is preceded by an 8-byte timestamp indicating the time of its creation. The timestamp is required for the merge process and is retained after the protection log is merged and is written to the sequential data set.

This section covers the following topics:

Merging Protection Logs

For recovery processing, all protection log data sets are automatically merged into a single log stream for each cluster database when an ADARES PLCOPY is executed in an Adabas Parallel Services nucleus cluster environment. The PLCOPY process accesses the parallel participant table (read ***Parallel Participant Table (PPT)*** elsewhere in this chapter) to determine which protection logs to copy and opens the appropriate data sets using dynamic allocation (read ***Dynamic Allocation*** elsewhere in this chapter). PLCOPY copies and merges as much data as possible; if a nucleus is still writing to a protection log data set, PLCOPY partially merges the data set.

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.

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. Records beyond this point are written to an 'intermediate' data set, which must be supplied as input to the subsequent merge.

Intermediate Data Sets

It is necessary to supply two intermediate data sets to the PLCOPY merge process: one is used as input and the other as output. ADARES analyzes both data sets to determine which is which. A cross-check ensures that the correct input intermediate data set has been supplied; the utility will not continue if the correct data set has not been supplied. The input intermediate data set contains "leftover unmerged" data from the last run of the utility, without which the merged protection data would be incomplete.

User Exits for Merging Protection Logs

A user exit is necessary for merging protection logs automatically: user exit 2 (UEX2) for dual logs or user exit 12 (UEX12) for multiple logs. However, you may choose to run without the user exit. This may be advisable for testing purposes, for example. If one nucleus in a cluster has the user exit specified, then all other nuclei in the cluster must also supply the user exit. The user exit supplied must be the same across all nuclei.

Sample user exits (USEREX2P and UEX12) are provided that illustrate the necessary changes for the intermediate data sets/files. Once the Link (DD) statements for the PLOG files/data sets have been supplied on the session startup JCL, you do not need to supply them again for ADARES as these are opened using dynamic allocation. If the Link (DD) statements are supplied, they are ignored.

Caution: Sample user exits and programs and are not supported under any maintenance contract agreement.

If there is a problem with the user exit, the Extended Error Recovery logic allows you to disable the exit and load a new exit without bringing the nucleus down.

Command Logs

In order to log commands during a session, a nucleus requires a command log data set. Command logging is activated by setting the ADARUN parameter `LOGGING=YES`.

Dual or multiple command logging requires that you set ADARUN parameters correctly. For more information about the ADARUN utility, read your *Adabas Operations Manual*.

This section covers the following topics:

- Merging Command Logs Manually
- Merging Command Logs Automatically
- Dynamically Modifying the CLOGMRG Parameter Setting

Merging Command Logs Manually

You can copy the dual or multiple command logs into a single sequential data set for a nucleus using the default ADARES CLCOPY function. Then, using the output from the CLCOPY execution as input, you can use the ADARES MERGE CLOG function to create a single cluster-wide command log containing, in chronological order, all Adabas commands executed by any of the cluster nuclei in the time period covered by the log:

```
ADARES MERGE CLOG NUMLOG=n
```

where *n* is the number of command logs to be merged, but not more than 32.

The ADARES MERGE CLOG function takes as input the sequential command log data sets/files from each of the cluster nuclei that were produced by running CLCOPY, merges them and copies the resulting command log data set from disk to a sequential data set.

Merging Command Logs Automatically

For accounting or other tracking purposes, you may want to automate the CLOG merge process. If you are using dual or multiple command logging, you set the ADARUN parameter `CLOGMRG=YES`, and if the correct user exit is available (user exit 2 for dual logs or user exit 12 for multiple logs), CLOGs are merged automatically as part of the CLCOPY process in much the same way that PLOGs are merged as part of the PLCOPY process. The ADARUN parameter `LOGGING=YES` must also be specified.

Existing CLCOPY jobs must be modified to include intermediate data sets/files. The sample CLCOPY job ADARESMC illustrates the necessary addition of the intermediate data sets/files.

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

You can use the same exit for the CLOG merge process that is supplied for the PLOG merge process (see the sample exit USEREX2P or UEX12):

- When `CLOGMRG=YES`, use a single user exit 2 or user exit 12 for both PLOG and CLOG merging; both PLOG and CLOG data sets/files are dynamically allocated. Data sets/files specified in the user exit JCL are ignored.
- When `CLOGMRG=NO` (the default), CLOG data sets/files are *not* dynamically allocated and must be specified in the ADARES JCL; therefore, user exit 2 or user exit 12 must submit different CLCOPY jobs that are appropriate for the nucleus that calls the exit.

Note: If `CLOGMRG=NO`, you can manually merge CLOGs across a cluster.

Likewise, CLOG data sets/files with slightly modified names can be included in the user exit 2 or user exit 12 JCL used for other NUCIDs.

Dynamically Modifying the CLOGMRG Parameter Setting

The `CLOGMRG` parameter setting can be changed dynamically with the new operator command `CLOGMRG`, the new ADADBS OPERCOM CLOGMRG function, or with the Modify Parameters function of Adabas Online System.

`CLOGMRG` is a global (that is, cluster-wide) parameter. A change made to one nucleus is automatically propagated to all nuclei in the cluster.

Coordinated Log Switching

The log merge process (PLOG or CLOG) makes it necessary to coordinate the log switching process so that the merge process can copy as much data as possible without causing too many switches.

The first cluster nucleus that needs to switch logs (when a log is full or the FEOFPL command for PLOGs or the FEOFCL command for CLOGs has been issued) sends a broadcast message to the other nuclei in the cluster indicating that it is switching. It then goes on as normal, switching and calling the appropriate user exit.

Note: User exit 2 for dual logs or user exit 12 for multiple logs is recommended and is required if PLOGRQ=FORCE is set. If it is supplied to one nucleus in a cluster, it must be supplied to all the other nuclei. PLOGRQ must have the same setting across all nuclei in the cluster. If running with PLOGs, only dual or multiple PLOGs are allowed.

When a cluster nucleus receives the message that another nucleus is switching logs, it must decide if it will switch based on the current status of its current log and whether it has an available log data set:

- If it switches, it does so without calling user exit 2 (dual logs) or user exit 12 (multiple logs).
- If it does not switch, it forces a log write of the current block so that ADARES has as much data as possible.

ADARES briefly waits at the start to give all nuclei time to switch. If a nucleus does not switch logs, ADARES copies as much data as possible from the active log (partial merge).

This section covers the following topics:

Global FEOFPL and FEOFCL Commands

The GLOBAL option has been added to both the ADADBS OPERCOM FEOFPL and FEOFCL commands:

- When the GLOBAL option is not specified, consideration is given to how full the PLOG or CLOG is before an attempt is made to switch them.
- When the GLOBAL option is specified, a log switch is forced for each nucleus in the cluster if there is an available log data set to write to. This may be useful in situations where all nuclei need to switch at the same time to capture all merged records in the output data set.

If a log data set to write to is not available, the switch does not occur even if GLOBAL has been specified. This functions the same way when the FEOFPL or FEOFCL command is issued for a single nucleus.

11 Parallel Participant Table (PPT)

The parallel participant table (PPT), located in a database's Associator, tracks all active Adabas nuclei in the cluster. As each nucleus becomes active, it registers itself in the PPT. Each PPT entry can be modified by the nucleus or by ADARES.

Acquiring the PPT

The PPT is an area of 32 contiguous blocks that the ADADEF utility allocates in the Associator when it defines a new database.

Registering Nuclei during Session Initialization

During session initialization as each Adabas cluster nucleus becomes active, it registers in the PPT based on the setting of its ADARUN `NUCID` parameter. The nucleus chooses the first of the 32 PPT blocks that is not yet assigned to another NUCID. If this is not the first run of this nucleus, the nucleus reuses its previously used PPT block. The relative block number of the chosen PPT block within the PPT (a number between 1 and 32) is called the *internal nucleus ID*.

Note: A noncluster nucleus always occupies the first block of the PPT and its NUCID is always zero.

Registering PLOG, CLOG, and Work Data Sets for Each Nucleus

Each active nucleus writes information to the Work, protection log (PLOG), and command log (CLOG) data sets that it is using for that session. The internal nucleus ID is also written to the PLOG records.

As each Work, PLOG, and CLOG data set becomes active, it is registered in the PPT. Information about these data sets is logged in the PPT entry of its nucleus.

When the PLOG or CLOG is merged, its PPT entry is updated with the information. ADARES and the cluster nuclei synchronize their PPT updates against one another to ensure that only one modification occurs at a time.

Work data set, protection log, and command log information remains in the PPT, even when CLOGs or PLOGs are merged or the session terminates.

Nucleus Termination Processing

If the nucleus terminates normally, the PPT entry is retained and marked as inactive. If a nucleus terminates abnormally, its entry in the PPT remains unchanged until the recovery process is complete.

Using PPT Information

The information about each Adabas nucleus maintained in the PPT is used when ADARES PLCOPY is copying and merging all active protection logs in the cluster.

It is also used during nucleus initialization to ensure that the data set/file information required for autorestart and PLOG merge is available:

- If the previously used Work data set or file and PLOG data sets or files are already in use by another nucleus, the nucleus will not start.
- If a different Work data set or file is provided and the original Work data set or file contains autorestart information, the nucleus will start but a warning will be printed.
- If different PLOGs or no PLOGs are provided and the previous session had PLOGs still to be copied, the nucleus will not start until those PLOGs have been copied/merged if `PLOGRQ=FORCE` is specified. If `PLOGRQ=FORCE` is not specified, the nucleus will start but a warning will be printed.

PPT Unavailable

In the case where the PPT becomes unavailable for some reason, the new parameter `NOPPT` is added to the ADARES PLCOPY function for emergency use. It specifies that the PPT is to be ignored and that the PLOG data sets of all cluster nuclei are to be taken from the JCL.

If you need to merge CLOGs while the PPT is unavailable, run a normal CLCOPY and then use the CLOG MERGE function to merge the sequential data sets.

SBLKNUM Parameter for ADARES PLCOPY NOPPT

The parameter SBLKNUM for the ADARES PLCOPY NOPPT function can be used in an emergency case when a last set of PLOGs must be copied and merged after the parallel participant table (PPT) in the Associator has been destroyed. In this case the ADARES utility does not know the last serial block number of the sequential PLOGs produced so far. By default, it starts numbering the blocks at 1 on the next sequential PLOG data set. If this sequential PLOG is concatenated to its predecessor in a REGENERATE function, ADARES would report an error in the PLOG block number sequence. The SBLKNUM parameter can be used to specify the correct starting block number for the sequential PLOG produced by the PLCOPY NOPPT function.

There are two ways to copy and merge the PLOGs when the PPT has been destroyed:

1. Without using the SBLKNUM parameter.
   - Run ADARES PLCOPY NOPPT, specifying all direct-access PLOG data sets of all cluster nuclei as DDPLOGnn data sets for the ADARES utility. ADARES copies and merges the relevant PLOGs and produces a sequential PLOG data set starting with PLOG block number 1.
   - In case the database or individual files need to be restored and regenerated, run the regenerate in two (or more) job steps. The first REGENERATE should cover all sequential PLOGs produced by PLCOPY functions without the NOPPT parameter; these PLOGs may be concatenated into one sequential input. If the REGENERATE is performed with ET logic (REGENERATE database or with CONTINUE parameter), specify the NOAUTOBACKOUT parameter. The second REGENERATE should cover the sequential PLOG produced by the PLCOPY with NOPPT.
2. Using the SBLKNUM parameter:
   - Determine the highest PLOG block number written so far by the regular PLOG merge processes. This number can be determined from an up-to-date checkpoint listing, from an ADARAI LIST protocol, or from the job protocol of the most recent PLCOPY execution.
   - Run ADARES PLCOPY NOPPT, specifying all direct-access PLOG data sets of all cluster nuclei as DDPLOGnn data sets for the ADARES utility. Specify the SBLKNUM parameter with value n+1, where n is the highest PLOG block number written so far, as determined in the previous step. ADARES copies and merges the relevant PLOGs and produces a sequential PLOG data set starting with the specified PLOG block number.
   - In case the database or individual files need to be restored and regenerated, no special arrangements need to be made to run the REGENERATE in separate job steps.

12 Dynamic Allocation

If used, protection log and command log data sets and files for use in Adabas Parallel Services cluster environments are specified in the session startup JCL. Session initialization writes the log data set names to the PPT and ADARES uses dynamic allocation to access them.

The same is true of the Work data set or file: it is specified in the session startup JCL. Session initialization writes the Work data set or file name to the PPT and, in the event of an autorestart, a nucleus is able to access the Work data set or file of a peer nucleus using dynamic allocation.

Dynamic allocation is currently available only in cluster environments (when the NUCID is greater than zero). However, dynamic allocation is used by a noncluster nucleus to determine that the previous nucleus session was a cluster session and PLOGs or CLOGs remain to be copied.

JCL Considerations

Files that are eligible for dynamic allocation by another database must be catalogued, rather than being accessed using explicit UNIT and VOLSER keywords in the JCL. If JOBCAT or STEPCAT DD statements are used to define a catalogue search structure, these statements must be used in a consistent way on all database startup jobs in the cluster.

13 Components Summary

Component	Number
ADASVC(ADARER)/SVCCLU	One or more per operating system image
ADACOM	One or more per operating system image
PPT	One per cluster; located in the database Associator
Adabas DBIDs	One per cluster; a practically unlimited number per operating system image
global cache structures	One per cluster; specified in each nucleus
global lock structures	One per cluster; specified in each nucleus
Adabas nuclei	Up to 31 per cluster

14 Switching Between Cluster and Noncluster Modes / PLOG Handling

Switching from cluster to noncluster mode (or vice versa) is possible only after normal termination. A starting nucleus checks in the PPT whether the previous session ended abnormally with a pending autorestart. If this is the case and the previous nucleus ran in the same mode as the starting nucleus, the session autorestart logic will be executed. If the previous nucleus ran in a different mode than the starting nucleus, the session start will terminate with an error.

The following sections illustrate a few scenarios where a cluster nucleus starts after the normal termination of a noncluster nucleus. PLOGRQ is not set to FORCE. These scenarios apply to two PLOGs as well as up to eight PLOGs.

Scenario 1

The previous session was noncluster mode, there are remaining PLOGs to be copied, there is no UEX2/12 in use, and the PLOG data sets are different from what was used in the previous session. The results of this scenario are as follows:

- The information in the PPT entry of the noncluster nucleus remains, and the new entry of the cluster nucleus is written.
- Initialization continues.

Scenario 2

The previous session was noncluster mode, there are remaining PLOGs to be copied, there is no UEX2/12 in use, and the PLOG data sets are the same as what was used in the previous session of a noncluster nucleus. The results of this scenario are as follows:

- A warning that the PLOG is being overwritten will occur and the PLOG flag in the previously used PPT block will be reset or the PPT entry will be overwritten (whichever is appropriate).
- Initialization continues.

Scenario 3

The previous session was noncluster mode, there are remaining PLOGs to be copied, UEX2/12 is in use, and the PLOG data sets are different from what was used in the previous session of a noncluster nucleus. The results of this scenario are as follows:

- UEX2/12 is called to submit a PLCOPY job that will copy and merge the PLOGs. The information in the PPT entry of the noncluster nucleus remains, and the new entry of the cluster nucleus is written.

- Initialization continues.

Note: If ADARES detects that there is data to be copied both from a cluster nuclei and from a noncluster nucleus (different PLOGs), it will copy the oldest data first.

15 Adabas Parallel Services and Other Software GmbH products

A complete list of the release numbers of Software GmbH products that support databases in Adabas Parallel Services clusters is provided in *Adabas Product Support* , in the *Adabas Parallel Services Release Notes*.

This chapter provides additional information on the interaction between Adabas Parallel Services and other Software GmbH products.

Adabas Online System

Adabas Online System communicates with all nuclei within an Adabas Parallel Services cluster and with all Adabas Parallel Services clusters on the operating system image. It includes functions related to Adabas Parallel Services.

Adabas Caching Facility

Software GmbH's Adabas Caching Facility can provide a performance boost to the Adabas cluster. It augments the Adabas buffer manager by reducing the number of read "execute channel programs" (EXCPs) to the database so that the available operating system facilities can be used without monopolizing valuable virtual memory resources.

Note: Write EXCPs are always issued to maintain the integrity of the database.

Under z/OS, Adabas Caching Facility provides support for 64-bit virtual storage.

Adabas Delta Save Facility

Software GmbH's Adabas Delta Save Facility (DSF) offers significant enhancements to ADASAV utility processing by backing up and restoring only the changed (delta) portions of Adabas databases.

DSF is intended for Adabas sites with one or more large, heavily updated databases that need to be available most of the time. Especially for sites where the volume of data changed on a day-to-day basis is considerably smaller than the total database volume, DSF provides for

- more frequent saves without interrupting database availability;
- enhanced "24 x 7 x 52" operation;
- full offline saving in parallel with the active database; and
- shorter REGENERATE duration during recovery.

Adabas Fastpath

Whenever possible, Adabas Fastpath reduces CPU consumption and increases throughput for Adabas systems by processing Adabas calls to completion in the client process. Database processing is thus avoided and capacity is increased for the whole operation.

Fastpath optimizes two types of database calls: direct access and sequential access. The results of direct access calls to the database are saved in a cache and repeat calls are satisfied from the cache rather than directly from the database. Read-ahead optimization routines are applied to sequences of commands to reduce redundant activity and thus accelerate the sequence.

Adabas Fastpath seamlessly optimizes database processing across multiple nuclei in one or more operating system images.

Adabas Vista

Adabas Vista is used to partition an Adabas file into multiple, separate files, each containing a part of the original larger whole. Partitioning makes it possible to distribute a file across multiple volumes or computers based on a criterion such as region or date. Adabas Vista is also used to translate Adabas database and file numbers, which allows an application to remain independent of the underlying physical layer.

Adabas Vista supports Adabas Parallel Services environments.

Adabas SAF Security

Adabas SAF Security enhances the scope of standard security packages based on the System Authorization Facility (SAF) such as RACF, CA-ACF2, and CA-Top Secret to encompass Adabas resources. It integrates Adabas into a central security repository and enables you to derive maximum benefit from your investment in that repository.

Adabas SAF Security adds protection for Adabas resources in Adabas Parallel Services environments.

Adabas Review

Adabas Review monitors the performance of Adabas environments and the applications executing within them. You can use information retrieved about Adabas usage when tuning application programs to achieve maximum performance with minimal resources.

Adabas Transaction Manager

The Adabas Transaction Manager (ATM), a selectable unit of Adabas, is a server for coordinating "distributed transaction processing" (DTP) in distributed Adabas environments. It manages global transactions that are distributed across multiple Adabas databases by coordinating changes to the databases in a seamless, integrated way, using a two-phase commit protocol when necessary.

At any time, ATM can account for in-flight transactions, suspect transactions, participating databases, and more.

ATM addresses two basic needs of the enterprise object revolution:

- the need to deliver industrial strength enterprise objects for widespread commercial use in mainstream, critical business systems.
- the need to spread the masses of data that Adabas customers manage more evenly across the computer(s) and organization.

ATM includes an online administration system based on Natural and available through Adabas Online System.

ATM provides limited support for Adabas Parallel Services. For more information, read ***Limited Distribution Transaction Support***, elsewhere in this guide.

16 Restricted Support for Adabas Features

This section describes facilities of Adabas that are not supported by cluster nuclei running under this Adabas Cluster Services or Adabas Parallel Services version. The facilities are supported normally for noncluster nuclei; however, no Adabas Cluster Services or Adabas Parallel Services functionality is available to them.

For an Adabas nucleus running in cluster mode (CLUSTER=LOCAL or SYSPLEX), the following features are not available and cannot be specified:

- `MODE=SINGLE`
- sequential protection log (DDSIBA)
- synchronous buffer flush (LFIOP=0)

The following features are not currently supported by nuclei running in cluster mode under this Adabas Cluster Services or Adabas Parallel Services version, but may be supported in subsequent versions of the product:

- `READONLY=YES` (receives PARM ERROR 71 if attempted).
- `UTIONLY=YES` can be specified for a cluster nucleus; if you start cluster nuclei with conflicting settings of `UTIONLY`, the system will change them to conform to the setting of the first active nucleus. Currently, however, the `UTIONLY` setting cannot be changed using an ADADBS OPERCOM or Adabas Online System function. Once the cluster is started, the only way to change the `UTIONLY` setting is to bring down the whole cluster and restart it with a different setting.
- online reorder.

Enhanced error recovery is supported; however, option changes are effective only for the local nucleus.

TCP/IP direct links are supported; however, the IP address/port is tied to an individual nucleus.

Triggers and Stored Procedures Facility

The triggers and stored procedures facility (SPT) is fully supported in a cluster environment:

- The ADARUN `SPT` parameter is global and must be set the same on all active nuclei.
- When a REFRESH is executed, the trigger table is passed to all active nuclei.
- Only the first nucleus to initialize reads the trigger file to create the trigger table; all subsequent nuclei obtain the trigger table from one of the already active nuclei.

Limited Distributed Transaction Support

Adabas Parallel Services can in a limited way participate in distributed Adabas transactions co-ordinated by the Adabas Transaction Manager, as follows:

- In each distributed transaction, a single cluster database is allowed to participate. Different cluster databases can participate in different distributed transactions.
- All nuclei of a cluster database participating in distributed transactions must be run with the new ADARUN parameter setting `DTP=ET`.
- Cluster databases can participate in distributed transactions only if all resource managers involved in these transactions are Adabas databases running with `DTP=RM`. `DTP=ET` cannot be supported when an external transaction framework is included in the transaction (CICS/RMI or RRMS).

Full support by cluster databases for distributed transactions will be provided in a subsequent release of Adabas Parallel Services.

User Table Cleanup

The operator command and AOS function CLUFREEUSER is available for situations where leftover User Table Elements in common storage need to be cleaned up. This function deletes such User Table Elements according to specified criteria.

By default, the cluster nucleus receiving this function deletes all User Table Elements assigned to itself which it does not know of (i.e., for which it has no User Queue Elements). If the GLOBAL option is specified, all User Table Elements are deleted that are assigned to nuclei that are no longer active, or that have no associated User Queue Elements in their assigned nuclei. For more information about the User Table Cleanup, see *Maintain the User Table* in the *Adabas Parallel Services Reference Guide*.

17 Getting Started

Nuclei start "open" to new users. To modify the status of nuclei throughout the cluster, read about the ADACOM command *SN - Set Nucleus Status* in the *Adabas Parallel Services Reference Guide*.

NUCIDs and DBIDs must have unique values in the range 1 through 65000.

ADACOM Starts

The ADACOM task must be run to set up the Adabas Parallel Services cluster environment before any cluster nucleus is started. ADACOM must be maintained in operation until the last cluster nucleus it manages is terminated.

A common storage area is allocated for maintaining information about active users and the nuclei to which they are assigned.

First Cluster Nucleus Starts

When a nucleus with a nonzero NUCID starts, it determines whether its ADACOM is running. If not, it will not start.

If its ADACOM is running, the first nucleus

- builds an ID table header (IDTH) element for the DBID, which marks the database as a cluster database. The DBID must be unique across the image in the range 1-65000. If `LOCAL=NO` is in effect, the DBID must be unique across all Entire Net-Work target IDs.
- sets an ID table entry (IDTE) to its NUCID in the nucleus table in CSA space to indicate that it is active. The NUCID must be unique across the image in the range 1-65000.

Additional Cluster Nuclei

If another nucleus for the same Adabas Parallel Services cluster comes up, the global DBID is already set and it sets its nucleus IDTE equivalent to its NUCID.

Index

S

T

U