Adabas Vista

Adabas Vista Programming Guidelines

Version 8.6.1

September 2025

ADABAS & NATURAL

Table of Contents

Preface

This document provides information related to programming applications for use with Adabas Vista.

The following topics are provided:

Using Adabas Vista with Adabas
Additional Programming-Related Topics
Adabas Vista Application Programming Interface
Using Adabas Vista with Cluster Services
Client Session Memory Requirements
Using Adabas Vista with Unicode
Using Extreme partitioned files
Using stored procedures and triggers

1   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.

2 Using Adabas Vista with Adabas

This section provides programming guidelines for using Adabas Vista with Adabas:

Adabas Vista ISN

If an Adabas file is defined as a partition belonging to a Standard or Extreme partitioned file, each Adabas ISN is modified before returning the ISN to the application. Such a modified ISN is called an Adabas Vista ISN. Correspondingly, any ISN received from an application program for such a file is interpreted as an Adabas Vista ISN.

- Standard Partitioning
- Extreme Partitioning

Standard Partitioning

For Standard partitioned files each Adabas ISN is modified with a partition identifier before returning the ISN to the application.

At the time the partition is defined, the space for a partition identifier is reserved within the 4-byte ISN field. The size of this area is based on the Adabas Vista file parameter `Maximum Number of Partitions`.

For example, the format of an Adabas Vista ISN for a standard partitioned file defined with a `Maximum Number of Partitions` set to 255 is as follows:

Offset	Length	Contents	Meaning
00	1	X'01' - X'FF'	The ID of the partition from which the record was retrieved.
01	3	X'000000' - X'FFFFFF'	The Adabas ISN of the record within the physical partition

Application programs that manipulate ISNs must be capable of handling the full 4-byte binary value returned. For Natural applications, any variable assigned to an ISN value should be defined with a minimum of P10. A value lower than this reduces the number of usable partitions and may result in a NAT1305 error code.

Caution: Special care is required when an application uses an ISN value as data. If a partitioned file is re-partitioned, the stored ISN may well become inaccurate.

Caution: When converting an Adabas file to a partitioned file, any ISNs for the file that may have been stored as data in other files must go through a conversion process to change the ISN values stored into correct Adabas Vista ISN values.

Extreme Partitioning

For Extreme partitioned files the application cannot infer any relationship between an Adabas Vista ISN and the Adabas ISN it represents.

ISN values relating to Extreme partitioned files are transient in their nature and therefore applications must not infer any persistent meaning to such ISN values.

For general information on Extreme partitioned files, refer to the section *Using Extreme partitioned files*.

For specific cautionary information regarding ISNs for Extreme partitioned files, refer to the **General Considerations** section in *Using Extreme partitioned files*.

Adabas Commands

E1 Command (Refresh File)

If an `E1` command is issued with an ISN value of zero and with a blank Command ID, the file is refreshed. This is also true for a partitioned file.

> **Note:** The `E1` command will be issued to each partition in turn. Any error which occurs when a later partition is refreshed (for example, Adabas response code 114 because the partition was not loaded with `PGMREFRESH=YES` specified) will result in a partially refreshed partitioned file.

S5 Command

The `S5` command is not supported for Partitioned Files.

Record Hold Logic

When using Adabas Vista, it is recommended to use the `GET (L4)` command to hold records immediately before modifying them. This is particularly relevant to those programs that would otherwise use the `READ (L6)` and `FIND (S4)` commands against a partitioned file.

Refer also to the section Distributed Lock Mode.

Read Sequential Direction Changes

When issuing an `L9` command sequence, a second direction change is not allowed for distributed access.

When processing a direction change during an `L9` command sequence, Adabas Vista places restrictions on format and length overrides in the search buffer.

Release Global Format IDs

A `RC` command which uses the C, L or O command options in order to release specific or all global format IDs will only be issued to the database specified in the command. In addition, the command options will always be changed to C to ensure that all global format IDs generated by Adabas Vista are also released by Adabas.

Optional ISN Positioning

Any program that accesses a partitioned file and requires the use of ISN positioning (for example, a starting ISN on an initial `L3` command, or a minimum ISN value on an initial Sx command) must ensure that the supplied ISN is in Adabas Vista format. Adabas Vista then targets the appropriate partitions of the partitioned file.

Command ID within a Search Expression

Adabas Vista does not support a Command ID value enclosed in parentheses as a search expression for partitioned file access.

Adabas Parameters

The following Adabas ADARUN parameters may need to be tuned for use with Adabas Vista.

Adabas Parameter	Description
LI	If a single file is partitioned into *n* files within the same database, it may obtain as many as *n* TBI elements per user as required for Adabas Vista distributed access. The increase in TBI elements depends on the amount of this distributed access and the number of partitions used in the average request. Therefore, the value of the `LI` parameter may need to be increased.
LQ	The `LQ` parameter is similar to the `LI` parameter above. The `LQ` pool may need to be increased.
LU	The `LU` parameter value should be set to 65535 whether or not Adabas Vista is used. This requires a minimum `NAB` parameter setting of 16.

Adabas Parameter	Description
NAB	The `NAB` parameter value may need to be increased if the Adabas Vista read-ahead feature is used and/or a significant number of partitions are defined to the database.
NC	If one or more partitions are added to a database for the first time, the value for the `NC` parameter may need to be increased.
NH	See the `NISNHQ` parameter below.
NISNHQ	Along with the `NH` parameter, this parameter may need to be increased if ■ a significant number of new partitions are added to the database and hold-based commands are issued becaused of distributed access. ■ the Adabas Vista read-ahead feature is used with hold-based commands.
NQCID	Similar to the `LI` parameter. The value for `NQCID` may need to be increased if a significant number of new partitions are added to the database and significant use of TBQ elements occurs because of distributed access.
NU	If one or more partitions are added to a database for the first time, the `NU` parameter value may need to be increased.
PREFETCH	This parameter is needed to control the Adabas Vista read-ahead feature at the job level. No other Adabas prefetch parameter is necessary.
TNAA, TNAE, TNAX, TT	Where partitions span more than one database, it is possible that a user may, during a series of related accesses, request numerous consecutive records from the same database within a distributed access. This may result in Adabas timing the user out on the other databases involved in the distributed access, which produces a response code 9 when Adabas Vista tries to continue the sequence on these databases. The values for the parameters `TNAA`, `TNAE`, `TNAX`, and `TT` should therefore be aggregated to the highest values for all databases containing partitions.

Adabas Security and Ciphering

When using Adabas Vista partitioned files, Adabas passwords and cipher codes must be the same for each partition.

ETID

We strongly recommend that the use of ETID is consistent (identical) across all databases used for a client session.

Caution: It is possible to dynamically change Vista translation rules (etc.) in the configuration file and apply the changes to a running system. Vista accommodates this as best as possible in running applications. For example, all active cursors (read loops, find loops, etc.) and

active transactions must be allowed to complete using the former set of rules (on a per session basis) before dynamic switchover to the new rules takes place. This covers most but not all situations. For example, there can be implications for handling ET data if rule changes alter the location where ET data are stored; unpredictable results are possible - especially if the rarely used situation of using ET data without setting ETID is used by the application. However, altering the location of the ET data store using only standard Adabas (without Vista) is very problematic since applications need to be fully coordinated to all look to the new location (only after all ET data has been moved), otherwise unpredictable results can occur, this is no different when Vista is used.

Transaction Directives

Adabas Vista generates transaction directives in accordance with standard distributed transactions making it compliant with the use of Adabas Transaction Manager. Correct distributed transaction processing is enforced by Adabas Vista (that is, an application is expected to be well-formed in the way it pursues transaction outcome). To explain...where multiple databases are modified the application is expected to pursue transaction outcome to all the changed databases, in series. For example, if two databases are modified then Adabas Vista expects to see two `ET` commands (one to each database) to cause a positive transaction outcome; or two `BT` commands (one to each database) for a negative transaction outcome. If an application tries to issue a transactional command without having fully issued all `ET/BT` commands for all databases in the previous transaction then RSP249 sub-code 496 will be issued. This behavior is compliant with distributed transaction processing standards, as implemented by Adabas Transaction Manager.

The runtime control `ET Data Database Number` is used to specify the location for transaction data.

Extended Hold

The P and M command options allow a program to commit (`ET`) or back out (`BT`) a transaction while leaving some records in hold status. Adabas Vista supports these options using the `Extended Hold` runtime control. Extended hold processing proceeds as follows:

■ `Extended Hold = Minimum` (default)

 ■ When Adabas Vista receives the first transaction directive, it has no knowledge of any P or M options that are to be applied to the databases involved in the transaction except those specified on this first directive.

 ■ Adabas Vista therefore honors the P or M option specified for the source database number of the first directive. It then generates the necessary directives for all other source database numbers without regard to P or M options. Thus, the client's transaction is committed or rolled back as requested, P or M options on the first directive honored and all other ISNs released off hold.

■ `Extended Hold = Maximum`

- When Adabas Vista receives the first transaction directive, it has no knowledge of any P or M options that are to be applied to the databases involved in the transaction except those specified on this first directive.

- Adabas Vista therefore honors the P or M option specified for the source database number of the first directive, but ensures that held records put on hold using other source databases are not released during the commit or roll-back process. Thus, the client's transaction is committed or rolled back as requested, but at this stage some of the transaction's held records may still be in hold status.

  As each subsequent directive (in the sequence) is issued, Adabas Vista honours the P or M option specified for those source database numbers. When the Vista proxy detects "end of directive sequence", the proxy generates the necessary directives for each source database number that was not party to the client's directive sequence.

- If the application program fails to terminate the sequence of directives, records may unintentionally remain in hold status in one or more target databases. In this case, it is necessary to intervene manually by operator command or Adabas Online System to stop the client and free its resources; otherwise, the resources are freed when the client's session times out.

CID Handling

Translation and partitioning features mean that CID commands may be re-directed to other databases/files which can cause inadvertent clashing of CIDs inside Adabas databases. Consequently the CID handling in Adabas Vista has now been strengthened to avoid this happening.

ET Data

If an ET or CL command with ET data is issued by a client and, as a result of this, Vista has to generate a series of such commands, then Vista will always issue the ET data on the first ET or CL of the series. This is in accordance with standard distributed transaction protocols making it compliant with the use of Adabas Transaction Manager.

The runtime control Database Number for ET Data is used to specify the location for transaction data.

⚠ **Caution:** It is possible to dynamically change Vista translation rules (etc.) in the configuration file and apply the changes to a running system. Vista accommodates this as best as possible in running applications. For example, all active cursors (read loops, find loops, etc.) and active transactions must be allowed to complete using the former set of rules (on a per session basis) before dynamic switchover to the new rules takes place. This covers most but not all situations. For example, there can be implications for handling ET data if rule changes alter the location where ET data are stored; unpredictable results are possible - especially if the

rarely used situation of using ET data without setting ETID is used by the application. However, altering the location of the ET data store using only standard Adabas (without Vista) is very problematic since applications need to be fully coordinated to all look to the new location (only after all ET data has been moved), otherwise unpredictable results can occur, this is no different when Vista is used.

Collation Descriptors

Adabas Vista does not support the use of collation descriptors for partitioned files.

Read Sequential Direction Changes

Changing the direction of an `L3` or `L9` sequence when the sequence has been previously subjected to multifetch is not allowed against a Partitioned File. This is consistent with the behaviour of `ADARUN PREFETCH` (refer to RSP021 sub-code 9 in the *Adabas Messages and Codes* documentation).

3   Additional Programming-Related Topics

This section provides additional information related to programming applications for use with Adabas Vista:

Coupled Files

Adabas Vista does not support search criteria that include syntax for physically coupled files or softly coupled files.

Command Limit

An optional commaned limit can be defined for a translation rule. If a client exceeds this limit then a response code 249, subcode 129 is returned. A command limit of 0 (zero) means that use of this particular rule is unlimited.

Error Handling

Adabas Vista uses the default Adabas response code 249 to indicate that an Adabas Vista processing error has occurred. This response code can be modified using the runtime control Error Response Code.

To qualify the error, a non-zero binary value is placed in the subcode field (the low-order 2-bytes of the Adabas Control Block Additions 2 field). Possible values for this subcode are provided in section Messages and Codes.

The application programmer can convert the response subcodes to error message text using the ERRM function of the Adabas Vista API (see **Application Programming Interface**).

When using Natural, the subcode may be retrieved using the USR0610N subprogram in library SYSEXT. This enables the following construct for an ON ERROR block:

```
0250   READ AVI1 LOGICAL BY NAME
0260       DISPLAY FIRST-NAME LAST-NAME SEX
0270   END-READ
0280   ON ERROR
0290   IF *ERROR-NR = 3249
0300      CALLNAT 'USR0610N' DB_ERR_STR
0310          WRITE 'AVI error, subcode:' DB_SUBCODE
0320   END-IF
0330   END-ERROR
0340   END
```

It is strongly recommended that error handling is implemented so that error conditions can be identified and handled immediately.

Refer to the runtime control `Error Reporting` for additional error reporting options.

Multiclient File Processing

Adabas Vista is able to process multiclient files. There are a few considerations that should be made when using this type of file with Adabas Vista:

■ Adabas Vista does not support superuser Owner IDs with distributed access. The sequence of records returned will not be the same as for a normal Adabas multiclient file. These superusers are most likely to be used for utilities, which are not normally run with Adabas Vista.

■ Adabas Vista cannot use the internal Owner ID as partition criteria. See the file parameter `Partitioning Field` for valid field types.

Shared Partitions

When processing Adabas Vista files that have some partitions shared, extra processing is required to determine the correct partition for a particular record, and to ensure that records are returned to the application in the most appropriate sequence for the command. This extra processing may have an adverse effect on performance for these command sequences.

For example, an `L1` command sequence processes all records in a file that is defined to Adabas Vista with three partitions, the first and third partitions sharing a single file. Adabas Vista will scan through the first file and extract all records that logically belong to the first partition. The second file will be processed as normal, and Adabas Vista will then scan through the first file again to extract the records that logically belong to the third partition.

When issuing an `L9` command against a shared partition where the descriptor used is not the partitioning field, although the count returned is correct, the ISN returned in the ISN Lower Limit field may indicate an incorrect partition. It is not possible to differentiate between the partitions in a shared file for an `L9` command.

Single Partition Focus

The Single Partition Focus feature exists only for backwards compatibility. You should change use of this feature to the ONLY option of the Access file partitioning parameter. The Single Partition Focus feature will be removed from future releases.

Updating the Partitioning Field

Special consideration must be made if modifying the record's partition criteria will result in the movement of the record from one partition to another. First, it is inevitable that the ISN will change – which is absolutely not a normal programming situation – and which may require some application design consideration. Second, moving from one partition to another may require a distributed transaction which will only be safe if you use Adabas Transaction Manager. Adabas Vista will allow such record movement but only using a specific exit that defines how the record movement is to take place. Please contact support in order to proceed in this area.

If no such exit is in use, any attempt to modify a record's partition criteria which would result in a cross partition move will return a response code. If such a modification is required, a DELETE and STORE operation must be performed.

4 Adabas Vista Application Programming Interface

The following Adabas Vista interfaces are made available to the application programmer, some of which allow pre-defined configuration settings to be overridden. The programmer should be aware that the most efficient mode of operation for Adabas Vista is to use the pre-defined configuration settings and that any dynamic runtime changes by API may incur additional overhead.

All API functions are available for Natural and most API functions are also available for 3GL programs.

Executing APIs with Natural and 3GL Programs

The Adabas Vista installation INPL dataset includes example Natural programs for each of the available API functions. These Natural objects must be installed according to the instructions provided in the section Installation.

In a 3GL environment, the supplied AVIPAPI program enables a 3GL program to utilise these API functions. Each API function available for use by a 3GL program has an example program in the Adabas Vista source library. These programs are named APIASM*nn*, where *nn* corresponds to the Natural example program number in the table below. For example, APIASM09 is the sample 3GL API program for the CATEGORY function.

> **Note:** API usage changes the normal behavior of the software. Once an API is used to change a setting, the new setting is usually persistent until another API (or a change by some other means) is used to alter the setting again.

API Function Overview

The following table lists the available API functions:

Function	Option	Description	Example
CATEGORY	UPDATE	Modifies the client's current target category.	API009UP
	EXTRACT	Extracts the client's current target category.	
CLOG	RESET	Resets the client commands trace area (only if runtime control Trace is active).	API001UP / AVICLOG
	-	Traces client commands (only if runtime control Trace is active). Show results newest-to-oldest.	
	+	Traces client commands (only if runtime control Trace is active). Show results oldest-to-newest.	
		"No option" has the same functionality as "+" .	
CONVISN	ADATOAVI	Converts an ISN from Adabas format to Adabas Vista format (including Partition ID).	API007UP

Function	Option	Description	Example
	AVITOADA	Converts an ISN from Adabas Vista format to Adabas format.	
CRITREP		Lists partitions identified as unavailable for the specified partition file.	API005UP
ERRM		Translates a supplied Adabas Vista subcode into an error text message.	API002UP
PAGE	UPDATE	Modifies the client's current Page list.	API003UP
	EXTRACT	Extracts the client's current Page list.	
PARTLIST		Lists all partitions for the specified partitioned file.	API006UP
PARTOPTS	UPDATE	Modifies the client's current `Access` and `Critical` parameter values for the specified partitioned file and partition.	API004UP
	EXTRACT	Extracts the client's current `Access` and `Critical` parameter values for the specified partitioned file and partition.	
TRTARG	UPDATE	Modifies the target details of the client's translation rule identified by the supplied source database and file number.	API008UP
	EXTRACT	Extracts the target details of the client's translation rule identified by the supplied source database and file number.	

API Function Descriptions

- CATEGORY Function
- CLOG Function
- PAGE Function
- PARTOPTS Function
- TRTARG Function

CATEGORY Function

A client may identify a target category override to Adabas Vista using the CATEGORY function. This replaces the target category established for the client by the active runtime controls. Adabas Vista will use this new target category to direct application calls to the corresponding target database and file number.

The use of the UPDATE option of this function will cause a transparent tidy-up resulting in any active cursors, etc. being ended.

CLOG Function

A client session may request tracing of commands using the CLOG API as long as the client runtime control TRACE is active.

For example, the results of the trace may be displayed oldest-to-newest:

```
1 OP 00000000    135     0   135     0    0 40404040 40404040      00000000
2 S1 00000000    135    77   135    77    0 000011BB 00000001      00000000
3 L3 00000001    135    32   135    32    0 0000026E 00000000   V  24150201
4 RC 00000001    135     0   135    32    0 00000000 00000000 S I 24150201
```

or newest-to-oldest:

```
4 RC 00000001    135     0   135    32    0 00000000 00000000 S I 24150201
3 L3 00000001    135    32   135    32    0 0000026E 00000000   V  24150201
2 S1 00000000    135    77   135    77    0 000011BB 00000001      00000000
1 OP 00000000    135     0   135     0    0 40404040 40404040      00000000
```

Furthermore, the whole log area can be cleared using the RESET option.

PAGE Function

A client may identify a page list override to Adabas Vista using the PAGE function. This replaces the page list established for the client by the active runtime controls. Adabas Vista will use this new page list to direct application calls to the corresponding target database and file number.

The use of the UPDATE option of this function will cause a transparent tidy-up resulting in any active cursors, etc. being ended.

PARTOPTS Function

The PARTOPTS function is used to update or extract the client's current `Access` and `Critical` partition parameter values for the specified partitioned file and partition.

The following Access values are available:

Value	Description
F	Set the `Access` partition parameter value to FULL.
R	Set the `Access` partition parameter value to READ.
N	Set the `Access` partition parameter value to NONE.
O	Set the `Access` partition parameter value to ONLY.
X	Reset all `Access` partition parameter values to the defaults.

The following Critical values are available:

Value	Description
Y	Set the `Critical` partition parameter value to YES.
N	Set the `Critical` partition parameter value to NO.

TRTARG Function

This function can be used to extract or update the target details of a supplied source database and file number. This source database and file number is used in conjunction with the client's current target category and page list to identify the corresponding translation target details. An update request will fail if the source file is currently involved in a transaction. This function is only available for individual file translation rules and not "all-files" translation rules.

> **Caution:** The use of the UPDATE option of this function means that the client session may be routing calls to a target database and file number which is different to that pre-defined in the configuration file.

5 Using Adabas Vista with Adabas Cluster Services

A clustered operating system is a collection of independent operating system images working together as one. The objective is to spread work around the cluster to achieve better load balancing, throughput, and availability.

Clustered applications appeared before their operating system counterpart. Application clusters operated over multiple processes within a single operating system image. Now clustered applications are enhanced to operate throughout an operating system cluster.

Adabas Cluster Services allows multiple instances of the Adabas server to operate against the same database across multiple operating system images, which enhances the general clustering approach for high-end systems.

Clustered Applications

A clustered application is a collection of independent jobs operating together as a single service.

A clustered application uses multiple jobs in unison to provide a single (clustered) service. CICS in MRO or plex mode, IMS TM, and UTM are examples of such clustered applications.

Clustered applications can route transactions dynamically from one job (member) in the application cluster to another, usually for load-balancing purposes. Dynamic transaction routing can occur within an operating system image, or across operating system images that are members of an IBM sysplex cluster.

The Adabas System Coordinator provides Adabas Vista support for dynamic transaction routing within and across operating system images.

Each implementation of a clustered application allows client sessions to migrate from one job to another across one or more images in an operating system cluster. The memory associated with a particular client session must be migrated along with the client session itself.

The following clustered applications may be implemented within a single operating system image:

- CICS/MRO with dynamic transaction routing in z/OS and VSE/ESA
- IMS/TM in z/OS
- UTM in BS2000

The following clustered applications are available for operating system clusters:

- CICSplex in z/OS and
- IMSplex in z/OS

Adabas Cluster Services

Adabas Cluster Services allows you to use multiple images of the Adabas server across an operating system cluster in order to provide truly scalable throughput.

Adabas System Coordinator

Support for clustered applications within a single system image or across multiple system images in an operating system cluster is enabled by the Adabas System Coordinator when running one or more of the following Adabas add-on products:

- Adabas Fastpath
- Adabas Vista
- Adabas Transaction Manager
- Adabas SAF Security

Adabas Fastpath also uses the Adabas System Coordinator to host the Asynchronous Buffer Manager service.

Client Runtime Controls for Clustered Applications

When defining runtime controls for CICS/MRO with dynamic transaction routing or for CICSplex, select the job type CICS Cluster.

Clustered application job types have user context information allocated from shared memory rather than from local (Job) memory.

Additional fields are available to support clustered applications:

Note: For non-clustered applications, leave these fields empty.

Field	Used to...
System Coordinator Group Name	link the clustered application to the relevant Adabas System Coordinator group.
Clustered Application Service Name	group together all instances of the application to the Adabas System Coordinator.
Manage Terminal Sessions or Manage All Sessions	determine the type of sessions managed by the Adabas System Coordinator daemon.

Refer to the *Adabas System Coordinator* documentation for more information.

6 Client Session Memory Requirements

The following table lists the main memory requirements for an Adabas Vista client session.

> **Note:** This table is not intended to be an exhaustive list of the memory requirements. It is recommended that you use this table during the initial analysis of a system's memory requirements. Once implemented, Adabas System Coordinator Online Services may be used to more accurately monitor the amount of real memory in use.

Memory requirement (per session) for each...		Size (in bytes)
session area (one only)		512
database accessed		+ 160
file accessed	Adabas file	+ 80
	Adabas Vista translation file	+ 80 + 64 + 48
	Adabas Vista Standard partitioned file	+ 80 + 64 + (48 + partitioning field length + partition concurrency * 5) * number of partitions.
	Adabas Vista Extreme partitioned file	As Standard partitioned file + 32K (TP system) OR + 1024K (Batch)
Command ID issued against	an Adabas file	+ 80
	an Adabas Vista translation file	+ 36 + 80
	an Adabas Vista partitioned file	+ (256 + 80 * number of involved partitions) + (length of Adabas buffers * number of involved partitions)
record held (within a transaction) against	an Adabas file	+ 64
	an Adabas Vista translation file	+ 64
	an Adabas Vista partitioned file	+ 64 * number of involved partitions

Example

1. Estimate the per session memory requirements for an application which has the following characteristics:

   ■ two databases

   ■ five Adabas files

   ■ two Adabas Vista translation files

   ■ one Adabas Vista Standard partitioned file

      ■ Partitioning field length is eight bytes

      ■ User partition concurrency is eight

      ■ Number of partitions is three

      ■ Majority of access is distributed (all three partitions involved)

      ■ Average length of Adabas buffers is 1000 bytes

   ■ Maximum Command ID concurrency of two (one Adabas file and one Adabas Vista partitioned file)

   ■ Average number of successful hold commands between `ET`/`BT` is 20

      ■ 18 against Adabas files

      ■ two against Adabas Vista partitioned file

Result

User session area:		512
Database requirements:	(2 * 160)	320
File requirements:	(5 * 80) + (2 * 192) + 1 * 432	1216
Command ID requirements:	80 + 3496	3576
Held record requirements:	18 * 64 + 2 * (64 * 3)	1536
	Total	7160

2. Multiply this value by the estimated number of application clients to estimate the application memory requirements.

3. If a single TP system is host to multiple applications, repeat this process for each application to provide an overall estimate of the TP memory requirements.

The required memory is requested from either

■ local TP memory using the host TP services; or

■ shared memory using the Adabas System Coordinator daemon.

The use of shared memory depends on whether the job is defined as a clustered application using job types such as CICS Cluster, IMS, or UTM.

> **Note:** It is strongly recommended that you discuss the use of such job types with your system programmer before you implement them.

7 Using Adabas Vista with Unicode

Adabas Vista supports the use of Unicode using wide-character fields (format W).

Where session encoding overrides are used, Adabas Vista expects them to be identical across all databases used during a client session; otherwise, unpredictable results may occur.

> **Note:** If partitioned files are being used, Adabas Vista does not support the use of distributed access by a descriptor, subdescriptor or superdescriptor with format W whose client has overridden the Adabas file encoding for wide-character fields.

> **Note:** All partitions of a (partitioned) file must be used with the same encodings.

8   Using Extreme partitioned files

This section details specific programming considerations when dealing with Extreme partitioned files.

Extreme file capacity

The mathematical maximum number of records for an Extreme partitioned file (across all the partitions) is determined by the maximum number of partitions multiplied by the maximum number of records per partition.

1. The maximum number of partitions is 255.

2. The maximum number of records per partition is 4,294,967,294.

Well-formed application logic

Good programming practices generally result in well-formed application logic. Here are examples of well-formed application logic.

A typical nested access connecting records of two files:

```
READ file-one BY abc FROM '1001'
            Some processing on the *ISN(file-one)
            FIND file-two WITH def = def(file-one)
                    Some processing on the *ISN(file-two) and perhaps also ↵
*ISN(file-one)
    END-FIND
END-READ
```

Add a modification into this (no END TRANSACTION is used to simplify the illustration):

```
READ file-one BY abc FROM '1001'
            Some processing on the *ISN(file-one)
            FIND file-two WITH def = def(file-one)
                    Some processing on the *ISN(file-two) and perhaps also ↵
*ISN(file-one)
                    UPDATE (or DELETE) *ISN(file-two)
    END-FIND
    UPDATE (or DELETE) *ISN(file-one)
END-READ
```

For the purposes of Extreme partitioned files, this logic is well-formed because the iterations deal with one record at a time, the *current* record.

In Natural programming this is typified by *ISN. Most application logic is programmed in this way, for no other reason than it makes sense because sensible application logic is cost-efficient to

maintain. Extreme partitioned files transparently support this logic because Adabas Vista maintains in memory partition details for the most recently used record(s).

Application logic that is not well-formed

In modern applications it is quite difficult to produce poorly-formed logic simply because the language syntax used today are themselves well-formed. Natural is an excellent example of that. However there are special cases where this may be necessary or there may be very old applications (usually 3GL) that are called where the logic is not so well-formed. These situations give rise to exceptions, which means you may have to adjust the application logic to adopt Extreme partitioned files (and perhaps even Standard partitioned files).

One classic example is where an application makes use of USERISN for a file. In this situation the application can use an ISN on a command that it has previously stored, in a file or in memory, meaning the command bears no relationship to any concept of *current* record – a clear indication of an application not being well-formed. There may also be other circumstances where an ISN-based command is used that conflicts with what would otherwise be considered the *current* record.

The current implementation of Extreme partitioned files does not support these exceptional circumstances, it may therefore be necessary to make small changes to the application in these (usually) relatively few areas. If such changes are not possible then contact support to discuss your requirements as it may be possible to provide APIs to support particular situations.

General Considerations

- ISN values relating to Extreme partitioned files are transient in their nature and therefore applications must not infer any persistent meaning to such ISN values. If such persistence is applicable then refer to section *Application logic that is not well-formed* for additional guidelines.

- Due to its transient nature, an application must not use an ISN value relating to an Extreme partitioned file as data.

- When converting an Adabas file to an Extreme partitioned file, any ISNs for the file that may have been stored as data in other files will be invalidated.

- ISN values relating to Extreme partitioned files that are supplied by an application must be ones that Adabas Vista itself has previously returned within the current client session and from within the same Adabas session.

- The CONVISN API is not currently supported for Extreme partitioned files.

9 Using Stored Procedures and Triggers

> **Note:** Most if not all these considerations apply to normal SPaTs usage without Adabas Vista too!

General Adabas usage

Commands issued by *participating* triggers affect the same Adabas client session as the originating client. It is the responsibility of the designer/programmer (of the originating client and participating trigger) to make sure Adabas usage does not collide in any way.

For example, it is possible for the originating client program and the participating trigger program to use the same partitioned, translated or ordinary Adabas file – but this means Adabas Vista will issue competing OP (or other) for the same session which will impact one or both programs causing unpredictable and error results.

The programmer must make sure all Adabas activity for these two peers of one session do not interfere with each other. Adabas Vista goes some way to help with this by making sure collisions do not occur within the database where the trigger fires ("the home database"), but Vista is unable to help with any other database access.

Transactional usage

As already explained commands issued by *participating* triggers (and the original client) act as the same Adabas session and may affect databases other than the home database. If these involve transactional commands then this means either or both of the two peers may have transactions in databases beyond the scope of the home database.

In industry terms this is a distributed transaction. This presents a problem in that only one of the two peers can control the completion of the transaction (commit or backout) which means the default action is that all the additional databases modified by the peer that does **not** control transaction completion are ignored causing unpredictable and error results – including data integrity issues.

Adabas Vista helps in some way by making sure the distributed parts of the transaction in the peer where the completion is issued are taken care of, in a serial way (multiple ET or multiple BT commands in series) but that does not take care of the problem on the other peer side.

The way to address this problem completely is to adopt usage of Adabas Transaction Manager because it is able to make sure all parts of the distributed transaction caused by both peers are completed safely. Please refer to the Triggers section of the Programmers' Guide in the Adabas Transaction Manager documentation for more information.

It is the responsibility of the designer/programmer to make sure the appropriate processing takes place. Some possibilities would be:

1. Deploy Adabas Transaction Manager so that distributed transactions can be freely used.

2. Make sure only the home database is modified even if Vista translation and partitioning is used by either or both of the peers.

3. Make sure only one of the peers modifies databases beyond the home database – and at least make sure Adabas Vista is there to propagate serial ET/BT commands automatically. This does not provide full data integrity (only Adabas Transaction Manager can accomplish this) but at least most of the time it will be okay because Adabas is so reliable.