Adabas System Coordinator

Adabas System Coordinator Operations and Programming Guide

Version 8.2.2

October 2017

ADABAS & NATURAL

This document applies to Adabas System Coordinator Version 8.2.2 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: COR-PROGRAMMING-822-20171008

Table of Contents

Preface

This document provides information related to Adabas System Coordinator operations and programming.

The following topics are provided:

Operational Guidelines

Starting the Adabas System Coordinator Daemon

Daemon Runtime Parameters

Daemon Operator Commands

Considerations and Configuration for using a Daemon

The Client Event Debug Monitor

Using the Unified Trace

User Queue Elements (UQEs) for Configuration File Access

Programming Guidelines

API To Retrieve Runtime Control Site Information

API To Modify Runtime Controls

1 Starting the Adabas System Coordinator Daemon

Normally, there is one Adabas System Coordinator daemon per operating system image.

Ensure that the database containing the Adabas System Coordinator configuration file is active before starting the daemon.

The daemon must be started before any TP monitors or batch jobs that use its services.

The daemon should run at a higher priority than the databases and client jobs that use it.

2 Daemon Runtime Parameters

The following parameters can be entered using DDCARD input. The PRODUCT parameter is mandatory. All other parameters are optional.

Parameter	Usage
CT	Command timeout limit.
FORCE	Overwrite ID table entry.
LOCAL	Define an isolated daemon.
LU	Length of intermediate user buffer.
MPMWTO	Display information messages.
NABS	Number of attached buffers.
NC	Number of command queue elements.
NXC	Number of sysplex command queue elements
PRODUCT	Identify the services to be made available. Mandatory.
START– Cold start feature	Delete shared memory from previous execution.

CT – Command Timeout Limit

Parameter	Use	Mimimum	Maximum	Default
CT	The maximum number of seconds (more precisely, units of 1.048576 seconds) that can elapse from the time a daemon request is completed until the results are retrieved by the sender through the interregion communication.	1	16,777,215	60

This parameter is used to prevent a request queue element (RQE) and attached buffer from being held indefinitely when a user with an outstanding request terminates abnormally.

Possible causes of a command timeout are

- address space is swapped out or cannot be dispatched;
- the task is cancelled or ABENDed;
- the task has low priority in a high-activity system.

FORCE – Overwrite ID Table Entry

Parameter	Use	Possible Values	Default
FORCE	Specify whether or not this daemon is to force an entry into the active node list.	YES | NO	NO

Possible values:

- `FORCE=YES`: Force an active entry, if one is available.

  `FORCE=YES` is usually not required. However, it may be needed if the previous daemon session ended abnormally, leaving the old entry in the active node list. Use this setting carefully.
- `FORCE=NO`: Causes an error if the node used by this daemon already appears in the active node list.

LOCAL - Define an Isolated Daemon

Parameter	Use	Possible Values	Default
LOCAL	Specify whether or not a daemon is to be isolated from other Entire Net-Work nodes.	YES | NO	NO

Possible values:

- `LOCAL=YES`: Isolates this daemon from other Entire Net-Work nodes.
- `LOCAL=NO`: The daemon can receive calls from other Entire Net-Work nodes.

LU – Length of Intermediate User Buffer

Parameter	Use	Mimimum	Maximum	Default
LU	Set the size of the intermediate user buffer area.	4000	65,535	65,535

The size specified must be large enough to accommodate all control information for commands passed to the node.

An error occurs if the `LU` parameter specifies a value greater than the byte count implied by the `NAB` parameter. If you change either parameter value, you may have to change them both.

MPMWTO – Display Information Messages

Parameter	Use	Possible Values	Default
MPMWTO	Specify whether or not to display information level (l-level) messages.	YES | NO	NO

By default, information level (I-level) messages are suppressed.

NABS – Number of Attached Buffers

Parameter	Use	Mimimum	Maximum	Default
NABS	Specify the number of attached buffers to be used.	0	500,000	16

An attached buffer is an internal buffer used for communication with the daemon.

For Adabas System Coordinator, this is an optional parameter that defines the number of attached buffers to be used for receiving requests from clients or from other daemon peers.

An attached buffer pool is allocated with a size equal to the value of the `NABS` parameter multiplied by 4096 bytes.

NC - Number of Command Queue Elements

Parameter	Use	Mimimum	Maximum	Default
NC	Set the maximum number of command queue elements.	20	32,767	100

The maximum number of command queue elements (CQEs) that can be processed simultaneously by this daemon.

NXC - Number of Sysplex Command Queue Elements

Parameter	Use	Mimimum	Maximum	Default
NXC	Set the maximum number of queue elements in the sysplex messaging command queue.	1	4096	256

This parameter is used only for multi-system daemons using XCF messaging, or for full Parallel Systems daemons using dynamic transaction routing (DTR). Although the default value will be sufficient for most sites, the number will dynamically expand as required when there is a high volume of XCF messages.

PRODUCT - Identify the Services to be Made Available

Parameter	Use	Possible Values	Default
PRODUCT	Specifies which product services are to be made available by Adabas System Coordinator: ■ AFP: Adabas Fastpath Asynchronous Buffer Manager ■ ATM: Adabas Transaction Manager ■ AVI: Adabas Vista daemon component ■ DTR: Dynamic Transaction Routing Service	AFP ATM AVI DTR	none

This parameter is used once for each service that is to be made available by the daemon.

Sites that use Adabas Fastpath will also require `PRODUCT=AFP`. Sites that use Adabas Vista or Adabas Transaction Manager and wish to support dynamic transaction routing in a clustered application will require `PRODUCT=AVI` and/or `PRODUCT=ATM`.

START– Cold start feature

Parameter	Use	Possible Values	Default
START	Normally, the daemon inherits and reuses shared memory allocations from a previous execution. Specify `START=COLD` to skip this recovery processing. **Note:** `START=COLD` is normally only used when advised by Software AG to avoid start-up problems related to auto-recovery from previous executions.	COLD	none

Parameter	Use	Possible Values	Default
	If there is a risk of daemons unnaturally terminating across an IPL then START=COLD should be used when restarting them.		

3 Daemon Operator Commands

The following operator commands are available through the z/OS `Modify (F)` command, VSE/ESA operator command, or BS2000 commands.

Command	Description
ADAEND	Terminates the daemon in an orderly manner. The daemon should not be terminated while there are jobs active.
HALT	The `HALT` operator command is only appropriate after ■ normal termination has already been requested via `ADAEND`. ■ normal termination is stalled. `HALT` causes the forced termination of the daemon by making it ignore the orderly shutdown of the daemon components (Fastpath, Transaction Manager, Coordinator, etc). This forced termination will result in all of the resources allocated by the daemon being unilaterally freed, including any and all memory shared between the daemon and client jobs. Any further ADABAS activity by connected client job(s) will result in unpredictable results due to the freeing of these resources. Software AG recommends `HALT` is only used in emergency situations when advised by Software AG because the results for clients that remain active will be unpredictable if they issue an ADABAS call subsequent to daemon termination.
DPARM	Displays the runtime parameters for this execution of the daemon.

DRES	Displays the allocated size, current usage and high-water mark for the following daemon resources:
	
	■ attached buffers
	  ■ Size is controlled by the `NABS DDCARD` parameter
	  ■ Current usage is not applicable
	■ command queue
	  ■ Size is controlled by the `NC DDCARD` parameter

Command	Description
	■ XCF command queue ■ Only applicable to z/OS daemons running in multi-systems XCF groups ■ Size is controlled by the NXC DDCARD parameter ■ Daemon threads ■ Current usage and high-water mark are not applicable ■ Shared memory ■ Size is controlled by the daemon "Shared memory area size (k)" parameter ■ Only applicable if the daemon "Shared memory area size (k)" parameter is not 0 ■ Shared threads ■ Only applicable for UTM and IMS services ■ For UTM, size is controlled by the UTM services "size (MB)" runtime control ■ For IMS, size is not applicable The information is also written to the daemon's output listing at daemon termination.
CAS DXCF	Displays message count information for cross-daemon XCF messages. The following counts are displayed in response messages CAS021I, CAS022I; ■ Msg-out: Total number of XCF messages sent by this daemon ■ Msg-in: Total number of XCF messages received by this daemon ■ Msg-rsp: Total number of message responses received ■ Msg-segs: Total number of message segments ■ Cq-Num: Number of queue elements in the Sysplex command queue. This will be equal to the command queue size specified, or defaulted, in the NXC parameter. ■ Cq-Hwm: Peak usage of the command queue. If this value is approaching the Cq-Num value it is advisable to increase the NXC value. ■ Cq-Full: Number of times a command queue full condition occurred. If greater than zero, increase the NXC value. ■ Cq-Post: Number of messages posted ■ RspPost: Number of responses posted This command is not required during normal daemon operations, but its use may be requested by Software AG personnel for diagnostic purposes. **Note:** XCF message counts will only be displayed if the daemon is running in "standard multi-system" or "parallel sysplex" mode.
CAS DXCF RESET	Displays message count information, as above, then resets all accumulated counters to zero.

4 Considerations and Configuration for using a Daemon

A daemon is required for Adabas Fastpath and Adabas Transaction Manager and is optional for Adabas SAF Security. There are other reasons for using a daemon:

As you will see, the daemon is a critical component and a failure can have a serious effect on the whole system. You can prevent a daemon failure by configuring your daemon jobs to use continuous operation.

Single-seat current activity displays

You can define jobs to send activity pulses to their local daemon, enabling current activities to be displayed for any appropriately defined job.

- Daemon configuration
- Client configuration

Daemon configuration

Single-seat current activity displays are only possible if the daemon is defined to use shared memory:

```
Run-mode: Daemon (node 2650)
Group: WORKSHOP   Daemon: ICFDCOR5   SVC: 254   Node: 2650    System: Multi
  Recovery
    Continuous Operation (Y/N).......: Y
  Daemon latency/pulse services
    Shared memory area size (k)......: 100000      Minimum (k): 0_______
           Dataspace name (if used)..: #COR5DSP
```

Here we specify that a 100 megabyte dataspace is to be used. If no dataspace name is specified, the memory is taken from ECSA for z/OS, ESVA for z/VSE or the specified common memory pool for BS2000. When using ECSA or ESVA a minimum size may also be specified.

The shared memory area size required is dependent on a number of factors: the products you have installed, the work profile of your applications, the number of sessions. As a rule of thumb, start off at 2k per session and monitor shared memory usage with the `DRES` daemon operator command.

If a session is unable to allocate shared memory, it will operate as normal but you will not be able to view its current activities.

Client configuration

You must also specify that activity pulsing is required in the client runtime controls of each job:

```
Activity pulse every.....: 5000___ commands or 60___ seconds
Group name..............: WORKSHOP   Daemon connection messages (Y/N): N
```

The above controls define how frequently sessions in the job should update their shared memory area (in this example every 5000 commands or 60 seconds) and also the group name of the local daemon to which activity pulses will be sent.

Single-system dynamic transaction routing

Adabas System Coordinator and its associated products need to maintain context information about client sessions. In some systems – UTM, IMS and CICS/MRO - client sessions can "jump" from one job to another and Adabas System Coordinator must ensure that the context information "jumps" with them. This is achieved by using daemon latency. This means that the daemon holds the context information for all sessions – either in daemon local memory, shared memory or on the COLAT disk file. For optimum performance and recoverability, Software AG recommends the use of shared memory and continuous operation.

Daemon configuration

The daemon should be defined to use continuous operation and shared memory:

```
Run-mode: Daemon (node 2650)
Group: WORKSHOP   Daemon: ICFDCOR5   SVC: 254   Node: 2650    System: Multi
  Recovery
    Continuous Operation (Y/N).......: Y
  Daemon latency/pulse services
    Shared memory area size (k)......: 600000      Minimum (k): 0________
           Dataspace name (if used)..: #COR5DSP
```

Here we specify that a 600 megabyte dataspace is to be used. If no dataspace name is specified, the memory is taken from ECSA for z/OS, ESVA for z/VSE or the specified common memory pool for BS2000. When using ECSA or ESVA a minimum size may also be specified.

The shared memory area size required is dependent on a number of factors: the products you have installed, the work profile of your applications, the number of sessions. As a rule of thumb, start off at 50k per session and monitor shared memory usage with the `DRES` daemon operator command.

If a session is unable to allocate shared memory, the daemon will use its local memory. However, daemon local memory is not recoverable in the event of a daemon restart and so its use should be avoided if possible.

Client configuration

Define the job as one of the DTR types, which enforce Daemon latency:

```
Type: CICS (DTR)   Name: WKS-DTR_
  Operation: Normal autodetect: X Enable without products: _ Disable all: _
  API runtime overrides....: N (Y/N)    Threadsafe operation...: N (Y/N)
  Use additional exits.....: N (Y/N)
  Maximum idle time (sec)..: 300_______ Non-terminal idle time.: __________
  Generate RSP009/79 (Y/N).: Y (until 0_________ seconds elapse)
  Messages - Local.........: Console Y and/or DDMSG file _
        Or - Daemon routing: _
  Latency - Local (Y/N)....: N

  Latency - Daemon (Y/N)...: Y
            to disk........: N
  Activity pulse every.....: _______ commands or _____ seconds
  Group name..............: WORKSHOP   Daemon connection messages (Y/N): N
```

and specify the group name of the daemon.

Multi-system dynamic transaction routing

Adabas System Coordinator and its associated products need to maintain context information about client sessions. In some systems – CICSPlex for example - client sessions can “jump” from one system image to another and Adabas System Coordinator must ensure that the context information “jumps” with them. This is achieved by using a daemon in every system image together with a common daemon latency file.

The latency file must be formatted using job CORI040 and should use Adabas device type 8390. The required file size depends on the products you have installed, the work profile of your applications and the number of sessions. As a rule of thumb, allocate 1 cylinder for every 10 sessions. This latency file must be allocated (as `COLAT`) to each daemon in the group.

- Daemon configuration

- Client configuration

Daemon configuration

The daemon group should be defined to use a disk file:

```
17:23:35                    Modify                      2015-02-06
                  System Coordinator Group              C11230M1

           Group Name: WORKSHOP      SVC ID: 254__

  System Type: _  Standard single-system image...
   (Mark one)     There is only one daemon in the group.
              X  Standard multi-system images - XCF...
                 This enables multiple XCF group daemons.
              _  Standard multi-system images - Net-Work...
                 This enables multiple Net-Work group daemons.

  Group wide latency service:
    Full crash recovery disk file (Y/N): Y




Command ==>
    PF1 Help         PF3 Exit       PF5 Upd        PF9 More
```

The daemon should be defined to use continuous operation. Shared memory is not required for daemon latency, however you may still define it to allow activity pulsing:

```
Run-mode: Daemon (node 2650)
Group: WORKSHOP   Daemon: ICFDCOR5   SVC: 254   Node: 2650    System: Multi
  Recovery
    Continuous Operation (Y/N).......: Y
  Daemon latency/pulse services
    Shared memory area size (k)......: 100000      Minimum (k): 0_______
           Dataspace name (if used)..: #COR5DSP
```

Client configuration

Define the job as one of the DTR types, which enforce Daemon latency:

```
Type: CICS (DTR)   Name: WKS-DTR_
  Operation: Normal autodetect: X Enable without products: _ Disable all: _
  API runtime overrides....: N (Y/N)    Threadsafe operation...: N (Y/N)
  Use additional exits.....: N (Y/N)
  Maximum idle time (sec)..: 300_______ Non-terminal idle time.: __________
  Generate RSP009/79 (Y/N).: Y (until 0_________ seconds elapse)
  Messages - Local.........: Console Y and/or DDMSG file _
        Or - Daemon routing: _
  Latency - Local (Y/N)....: N

  Latency - Daemon (Y/N)...: Y
            to disk........: Y
  Activity pulse every.....: ________ commands or _____ seconds
  Group name...............: WORKSHOP   Daemon connection messages (Y/N): N
```

and specify Y for “to disk” and the group name of the daemons.

5 The Client Event Debug Monitor

The client event debug monitor is used to troubleshoot problems with Adabas System Coordinator or with other Adabas products that work closely with it. Normally it is only used under guidance from Software AG Customer Support, however, as you will see by reading on, you might find it useful when troubleshooting your own systems too.

The debug monitor will write diagnostic information to a file (CORDUMP) based upon the settings that you make. By default, the debug monitor is inactive.

Considerations before using the Debug Monitor

Enabling CORDUMP

The CORDUMP file must be available to the job being monitored and sized according to the amount of output requested. In some operating systems (eg. z/OS, z/VSE), this can be done using the job's execution control script (eg. JCL, JCS). In other operating systems (eg. BS2000), it will automatically default to list output.

Additional information regarding the attributes of the CORDUMP file is available on the Adabas System Coordinator Debug Event Monitor Controls online help screen.

Runtime Overheads

The debug monitor has minimal CPU runtime overhead.

Event Reporting and Output

When a monitored event occurs, information is written to the CORDUMP file based on the output options set. Careful consideration should be given to the setting of these options to avoid excessive output being written. For some events it may be necessary to capture significant amounts of information, clients running in the monitored job may therefore experience a delay in processing during the capture of this information.

Refer to **Setting Debug Monitor Controls** for an explanation of all the debug monitor controls.

Monitoring Exceptions

Generally, the debug monitor can be used to report on the majority of events, however there are certain exceptions:

- Excluded Events
- CICS Threadsafe

Excluded Events

Some internal processing errors may be reported as a Response 101 with sub-codes which are not eligible for reporting by the debug monitor. If you are unsuccessfully trying to report on a sub-code for Response 101 please check the messages and codes for the particular sub-code to see if it is excluded from this feature.

CICS Threadsafe

The debug monitor is automatically disabled for CICS jobs running in threadsafe mode. In order to use the debug monitor for these jobs, set the Adabas System Coordinator client runtime control "Threadsafe operation" to "N" and then set it back after you have finished.

Setting Debug Monitor Controls

The debug monitor controls are part of Adabas System Coordinator's Client Runtime Controls.

In SYSCOR, modify your client runtime controls; press PF9 ('More') then select option 2 ('Debug Settings') which presents the following screen:

```
18:45:13     ***** A D A B A S   SYSTEM COORDINATOR 8.2.1 *****     2010-12-15
                     -  Debug Event Monitor Controls  -              U1SCJBM1

 Debug all sessions (Y/N) .......: Y     Maximum debug reports ......: _____
     Response code: ___ Sub-code : _____ or mark for generic monitor : _
     Optionally for database ....: _____ and file number ............: _____
   Additional debug monitor (Y/N), use only as directed by Software AG:
     System Coordinator .........: N     Adabas Transaction Manager .: N
     Adabas Fastpath ............: N     Adabas Vista ...............: N

   Report content in order of output amount, mark one:
     None .......................: X     Client session only ...........: _
     All sessions for the client : _     All sessions for the job ......: _
     All memory for the job .....: _
   Additional report content (Y/N):
     CIB ................: Y   CAB ..............: Y   ID table .........: Y
     Registers on entry : Y   TP areas .........: Y   Stack ............: Y




 Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Upd
```

A description of each parameter follows:

Parameter	Use	Values	Default
Debug all sessions (Y/N)	Defines the scope of the debug monitoring. When activated, the default, "Y", will result in all sessions being monitored immediately from the start of job. Specifying "N" will defer any monitoring until individual sessions have been activated. Refer to **Activating the Debug Monitor** for more information on this selective capability.	Y | N	Y
Maximum debug reports	Limits the number of times the monitored event will cause a report to be written to the CORDUMP file. The default, "0", means no reports will be written. Usually a value of 1 is sufficient for most events. **Caution:** Specifying an excessive value could severely impact system performance.	0 - 65535	0
Response code / Sub-code, or mark for generic monitor	Defines the event to be monitored: ■ A specific response code without sub-code ■ A specific response code and sub-code	Response code: 0 – 999 Sub-code: 0 – 65535	0

Parameter	Use	Values	Default
	■ A generic monitor for non-zero response codes (excludes 0, 3, 9, 148, etc. - a full list can be found in the help screens)		
Optionally for database / file number	From time to time Software AG may supply a diagnostic fix in order to generate additional reports to the CORDUMP file. These parameters provide control at the job level for such reports for the following products: ■ Adabas System Coordinator ■ Adabas Transaction Manager ■ Adabas Fastpath ■ Adabas Vista The default, "N", means no additional reporting. **Note:** These defaults should only be changed under guidance from Software AG.	Database: 0 – 65535 File: 0 - 65535	0
Additional debug monitor (Y/N)	From time to time Software AG may supply a diagnostic fix in order to generate additional reports to the CORDUMP file. These parameters provide control at the job level for such reports for the following products: ■ Adabas System Coordinator ■ Adabas Transaction Manager ■ Adabas Fastpath ■ Adabas Vista The default, "N", means no additional reporting. **Note:** These defaults should only be changed under guidance from Software AG.	Y | N	N
Report content (in order of output amount)	Defines the type of information generated each time a monitored event causes a report to be written to the CORDUMP file: ■ None ■ Client session only ■ All sessions for the client ■ All sessions for the job ■ All memory for the job The most commonly used option is 'All memory for the job' because this guarantees all available diagnostic information will be written.	See description	None

Parameter	Use	Values	Default
Additional report content (Y/N)	Each time a monitored event causes a report to be written to the CORDUMP file, the following areas are included by default (regardless of the report content parameter): ■ CIB ■ CAB ■ ID table ■ Registers on entry ■ TP areas ■ Stack There is usually no reason to change these defaults except under guidance from Software AG.	Y | N	Y

Activating the Debug Monitor

Activation at Job Start

At job start, if "Maximum debug reports" is > 0 and the debug monitor control "Debug all sessions" is "Y", then debug monitoring will be automatically activated.

Activation for an Individual Session

At job start, if "Maximum debug reports" is > 0 and the debug monitor control "Debug all sessions" is "N", then debug monitoring will be deferred until it is manually activated for a selected client session, as follows:

1. Display the summary of session information for a job. Refer to Display Session Information within Current Activity Displays for information on how to do this.

```
17:23:36 ***** A D A B A S   SYSTEM COORDINATOR 8.2.2 (I001) *****  2012-08-29
                      -  Display Session Information  -             C12130M1
 Run-mode: Pulsing (node 2660)                Perspective: Daemon (node 2660)
 Service: None     Job: DA2FCI23 C34504   ID: CI23     Commands:          34,711
 Sessions: 3         Session size: 80K     Work size: 76K     Total: 199K
 C     Session id    Login id     Number
    *_______                    of commands Area     System         Status      ↩

 _  TCM9    DA2FCI23 TEAM1              198                    running in Adabas
 _  TCN0    DA2FCI23 TEAM2            4,972                         at rest
 _  TCN1    DA2FCI23 TEAM3           23,489                         at rest









 Mark with any character to expand or T(asks)
End of List
 Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Refr                           JobID             Menu
```

2. Mark a session with a “T” to display the list of possible tasks:

```
+-------------------------------------------------------------------+
|  17:33:03           Client Session Tasks          2012-10-25  |
|  Job name: DAEFCI18 Session ID: TCCN    DAEFCI18              |
|                                                                   |
|  Select one task:                                                 |
|  _ Snap                                                           |
|  X Switch debug on/off                                            |
|  _ Purge                                                          |
|                                                                   |
|                                                                   |
|                                                                   |
|                                                                   |
|                                                                   |
|                                                                   |
|                                                                   |
|              PF3 - Exit    Enter - Perform task                   |
|                                                                   |
+-------------------------------------------------------------------+
```

3. Select the task “Switch debug on/off” to activate and deactivate the debug monitor for this particular session.

6 Using the Unified Trace

The unified trace is used to troubleshoot problems with the System Coordinator or with Adabas products that work closely with it. Normally it is only used under guidance from Software AG Customer Support. However, as you will see by reading on, you might find it useful when troubleshooting your own systems too.

The default is that it is inactive.

The unified trace can be activated in the client, daemon, or database based upon the settings that you make.

The Client Unified Trace

The client unified trace settings are defined in client runtime controls. They can be configured statically (through the Maintenance function of SYSCOR) or dynamically (through the Current Activity Display function of SYSCOR). For random problems experienced by your commercial users it may be necessary to configure the settings statically (subject to the Trace size warning – see below), however in some cases it is possible to reproduce the problem in controlled circumstances. In this case you can configure the settings dynamically for the specific client session(s).

To configure the required trace settings, modify your selected client controls in SYSCOR (either statically or dynamically); press PF9 then select option 3 (unified trace settings) which presents the following screen:

```
 16:42:40 ***** A D A B A S   SYSTEM COORDINATOR 8.2.2 (I001) *****  2012-08-23
                         -  Unified Trace Settings  -                U1SCTRM1

Trace collection settings
 Trace collection (Y/N)..............: N
 Local trace memory (k)..............: 0___    0=none,minimum=32,maximum=1024
 Trace recording settings
  Write a copy to local trace file....: _       Mark for local trace file
  Forward to the daemon trace file....: _       Mark for daemon trace file
  Synchronous writes..................: _       Mark to limit to synchronous
  Single writes activation point......: _____   No buffering after n commands
  Flush for significant session events: _       Mark to flush on CL (etc)
Dynamic options
 Limited debug trace:
  Response code trace activation.....: 0__
  Sub-code...........................: 0____
  Generic error response codes (Y/N).: N
  Debug event activation (Y/N).......: N



 Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Upd
```

- Trace collection
- Local trace memory
- Dynamic options

Trace collection

Trace data will be collected only when this parameter is set to Y and a local trace memory size has been specified. Once started, data is collected into the trace buffer and, when full, will wraparound overwriting the oldest content and maintaining the newest content. The period of time for which trace data remains in memory is therefore dependent on the size of the trace buffer. The content of the trace buffer can be displayed through SYSCOR Current Activity Display and for information on how to do this refer to Expand Adabas client sessions.

Local trace memory

This specifies the amount of memory (in k) to be used for the unified trace buffer and is allocated at the individual session level.

Caution: When statically defining the trace size for a job's runtime controls through the SYSCOR Maintenance function, due consideration must be given to the trace size as it may have a significant impact on the job's memory consumption. Such consideration is far less significant for a trace size specified dynamically to an individual session through the SYSCOR Current Activity Display function.

In addition to collecting and displaying the most recent trace activity, additional options are available for managing the data within the trace buffer:

Write a copy to local trace file

Note: This option is only selectable for job types Batch, TSO, and TIAM.

You can optionally choose to flush collected trace data to a local sequential file (COTRC). This file must be defined to the job's execution control script. If using COTRC you must add the Adabas load library to the job's loading environment.

Forward to the daemon trace file

Note: This option is only selectable for jobs that have been defined to use the COR daemon.

You can choose to flush collected trace data to a daemon trace file. Refer to **The Daemon Unified Trace** for information on how to implement a daemon trace file.

Synchronous writes

Note: This option is only applicable if "Write a copy to local trace file" or "Forward to the daemon trace file" has been selected.

By default, the flushing of collected trace data is done asynchronously. Select this option if you prefer this to be done synchronously.

Single writes activation point

Note: This option is only applicable if "Write a copy to local trace file" or "Forward to the daemon trace file" has been selected.

When a session's activity reaches the number of commands specified here, the current trace content will be flushed out and thereafter trace entries will be written one at a time as each one is completed. No buffering will occur.

Flush for significant session events

Note: This option is only applicable if "Write a copy to local trace file" or "Forward to the daemon trace file" has been selected.

Selecting this option will cause additional flushing of the collected trace data at predefined events (for example at CL command time).

Dynamic options

By far the most common event that needs to be traced is an unexpected Adabas response code. You can choose to stop further trace collection following one of these events:

- a specific response code without a subcode
- a specific response code with a specific subcode
- a generic response code (this control relates to all response codes except those which do not indicate an error of significance. For example, response codes 0, 3, 9 and 148, a full list can be found in the help screens).

Once collection has stopped, the trace buffer in memory remains displayable for as long as the session remains active. If you selected one of the options to additionally write to disk then the current trace buffer is also written out to disk for more permanent analysis.

The Daemon Unified Trace

The daemon unified trace settings are defined in daemon parameters. In SYSCOR, list the daemons in your daemon group then modify the appropriate daemon which presents the following screen:

```
15:21:11 ***** A D A B A S   SYSTEM COORDINATOR 8.2.2 (I001) *****  2012-08-24
              -  Adabas System Coordinator Daemon Parameters  -      C11261M1
 Run-mode: Pulsing (node 2650)
 Group: WORKSHOP   Daemon: ICFDCOR5   SVC: 254   Node: 2650    System: Multi
   Recovery
     Continuous Operation (Y/N).......: N
   Daemon latency/pulse services
     Shared memory area size (k)......: 10240___    Minimum (k): 0________
            Dataspace name (if used)..: _________
     Daemon memory area size (k)......: 10240___    Minimum (k): 0________


   Unified trace settings
     Trace collection (Y/N)...........: N
     Local trace memory (k)...........: 0___ (0=none minimum=32 maximum=1024)
     Use trace file (Y/N).............: N
     Wraparound trace file when full..: N
   Debug settings
     CORDUMP for transient situations.: N  Number of outputs: 0____


 Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Upd                                       Menu
```

Caution: *Daemon trace collection to the trace file quickly fills the file and can cause information forwarded by client jobs to be lost. This is why the default for the daemon trace is to be inactive, so that it can be used sparingly by activating it dynamically for short periods.*

- Trace collection
- Local trace memory
- Use trace file
- Wraparound trace file when full

Trace collection

Trace data will be collected only when this parameter is set to Y and a local trace memory size has been specified. Once started, data is collected into the trace buffer and (optionally, a trace file – see below). When the trace buffer is full, it wraps around overwriting the oldest content and maintaining the newest content. The period of time for which trace data remains in memory is therefore dependent on the size of the trace buffer. The trace can be activated (and de-activated), and its content displayed, through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Coordinator daemon nodes in Network Discovery. In addition to collecting and displaying the most recent trace activity, all trace data can be written to disk.

Local trace memory

This specifies the amount of memory to be used for the unified trace and is allocated at the individual thread level. This size can also be changed through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Coordinator daemon nodes in Network Discovery.

Use trace file

You can choose to write collected trace data to a local BDAM file (COTRC). This file must be defined to the daemon's execution control script. This option can also be changed through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Coordinator daemon nodes in Network Discovery.

When this option is selected, the following type of trace data is written to the file:

- Daemon collected trace data (Local trace memory must be non-zero and the trace collection dynamically activated)
- Client forwarded trace data (for more information refer to **The Client Unified Trace**).

Wraparound trace file when full

When selected, this option will cause trace data to wraparound overwriting the oldest content and maintaining the newest content. This option can also be changed through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Coordinator daemon nodes in Network Discovery.

The Database Unified Trace

The database unified trace settings share the same configuration as the daemon unified trace settings which are defined in daemon parameters. In SYSCOR, list the daemons in your daemon group then modify the appropriate daemon which presents the following screen:

```
15:21:11 ***** A D A B A S   SYSTEM COORDINATOR 8.2.2 (I001) *****  2012-08-24
              -  Adabas System Coordinator Daemon Parameters  -      C11261M1
 Run-mode: Pulsing (node 2650)
 Group: WORKSHOP   Daemon: ICFDCOR5   SVC: 254   Node: 2650    System: Multi
   Recovery
     Continuous Operation (Y/N).......: N
   Daemon latency/pulse services
     Shared memory area size (k)......: 10240___    Minimum (k): 0________
            Dataspace name (if used)..: _________
     Daemon memory area size (k)......: 10240___    Minimum (k): 0________


   Unified trace settings
     Trace collection (Y/N)...........: N
     Local trace memory (k)...........: 0___ (0=none minimum=32 maximum=1024)
     Use trace file (Y/N).............: N
     Wraparound trace file when full..: N
   Debug settings
     CORDUMP for transient situations.: N  Number of outputs: 0____


 Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Upd                                       Menu
```

The shared configuration is described here:

- Trace collection
- Local trace memory
- Use trace file
- Wraparound trace file when full

Trace collection

Trace data will be collected only when this parameter is set to Y and a local trace memory size has been specified. Once started, data is collected into the trace buffer and (optionally, a trace file – see below). When the trace buffer is full, it wraps around overwriting the oldest content and maintaining the newest content. The period of time for which trace data remains in memory is therefore dependent on the size of the trace buffer. The trace can be activated (and de-activated), and its content displayed, through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Adabas nodes in Network Discovery. In addition to collecting and displaying the most recent trace activity, all trace data can be written to disk.

Local trace memory

This specifies the amount of memory to be used for the unified trace and is allocated at the individual thread level. This size can also be changed through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Adabas nodes in Network Discovery.

Use trace file

You can choose to write collected trace data to a local BDAM file (COTRC). This file must be defined to the database's execution control script. This option can also be changed through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Adabas nodes in Network Discovery.

Wraparound trace file when full

When selected, this option will cause trace data to wraparound overwriting the oldest content and maintaining the newest content. This option can also be changed through SYSCOR Current Activity Display and for information on how to do this refer to the section on tasks for Adabas nodes in Network Discovery.

7 User Queue Elements (UQEs) for Configuration File Access

The vast majority of parameter settings for all Adabas client-based add-ons are defined in the configuration file. Therefore it is extremely easy to modify runtime settings without having to locate the JCL for all client jobs, databases, etc. that run in your system. This is a major advantage over embedding runtime settings inside JCL because client job JCL is mostly unavailable to the administrators to freely make changes. The configuration file must be accessed at runtime by Adabas System Coordinator client jobs during start up to pick up runtime settings. Adabas System Coordinator generates commands to the configuration file using special Adabas sessions (UQEs). This is now an advantage because in previous releases there was at least one UQE for each batch job step that ran in your systems (this is a very, very large number in most systems). Each of these UQEs would be created and deleted with very few commands issued for them. Now the identity of the UQE is set so that it equates to a job's process identity (for example: address-space identity in z/OS) and consequently these sessions are reused by all batch job steps that use the same process identity (regardless of job name). The result is that the total number of these UQEs is now (approximately) equal to the number of process identities in your system that are used for Adabas work (which is a much smaller number than before) - therefore the constant create UQE, delete UQE activity that went unnoticed inside Adabas in previous releases now doesn't happen at all.

8 API To Retrieve Runtime Control Site Information

Adabas System Coordinator allows storing of site-specific data in runtime controls. The contents and format of the data are entirely under your control. A typical use for it might be to define your own runtime controls (for example dynamic Natural parameters). The data can be retrieved using the supplied APIs. Two APIs are provided; one for 3GL and Assembler programs and one for Natural programs.

3GL API

The 3GL API is contained in the supplied COR3GLI load module. There are also some supplied source members showing how to use the API:

- APIINF01: example of using the API in environments other than CICS
- APIINF02: example of using the API in CICS
- COR3GLIA: a parameter data area for calling COR3GLI

To use the 3GL API:

1. Allocate storage for the parameter data area (1792 bytes).
2. Initialize the storage to binary zeroes.
3. Set the interface version (field name INFVRS in COR3GLIA) and function (INFFNC).
4. Under CICS, set the name of the Adabas link module to be used (INFCICN). The link module must be capable of accepting parameter lists via the COMMAREA. If not under CICS, INFCICN must contain binary zeroes or spaces.
5. If using the reentrant ADALNKR, allocate a modified area and set its address (INFAMOD).
6. Link this program together with COR3GLI and, if not under CICS, your Adabas interface module, for example ADAUSER.
7. After calling COR3GLI, INFRC will contain 0000 and INFDATA will contain the site information for this session; or, INFRC will contain a non-zero return code and INFRT will contain an explanatory message.

Natural API

The Natural API is contained in library SYSCOR, subprogram CORNATI. There are also some supplied source members in SYSCOR, showing how to use the API:

- APIINF-P: example of calling CORNATI
- CORNATIA: parameter data area for calling CORNATI

To use the Natural API:

1 Ensure that subprogram CORNATI is available, by copying it to your Natural library or adding SYSCOR to your library's steplibs in Natural Security.

2 Set CORNATI-VERSION and CORNATI-FUNCTION.

3 Call CORNATI.

4 After calling CORNATI, CORNATI-RC will contain 0000 and CORNATI-SITEINFO will contain the site information for this session; or, CORNATI-RC will contain a non-zero return code and CORNATI-RT will contain an explanatory message.

Return Codes

These are the non-zero return codes which may be set:

Return Code	Description
0001	Invalid interface version (must be 01)
0002	Invalid function (must be GETINFO)
0003	System Coordinator not available
0004	System Coordinator internal error
0005	System Coordinator internal error
0006	Adabas interface not linked (3GL API only)
0008	No site information for this session

9 API To Modify Runtime Controls

Adabas System Coordinator uses runtime controls to determine execution behavior. Applications can modify some of these runtime controls dynamically, via API. Two APIs are provided; one for 3GL and Assembler programs and one for Natural programs. Currently the only runtime controls that can be modified are the z/OS-only controls *Review* and *Client Monitor*.

3GL API

The 3GL API is contained in the supplied COR3GLI load module. There are also some supplied source members showing how to use the API:

- APIREV01: example of using the API in environments other than CICS
- APIREV02: example of using the API in CICS
- COR3GLIA: a parameter data area for calling COR3GLI

To use the 3GL API:

1. Allocate storage for the parameter data area (1792 bytes).
2. Initialize the storage to binary zeroes.
3. Set the interface version (field name INFVRS in COR3GLIA) and function (INFFNC).
4. Set the product code (INFCPROD) for which controls are to be modified (currently, this must be 'COR').
5. Set the names (INFCNAME) of the runtime controls to be modified ('REVIEW' and/or 'REVIEW-CLIENT'), together with the required value (INFCVAL, 'Y' or 'N').
6. Under CICS, set the name of the Adabas link module to be used (INFCICN). The link module must be capable of accepting parameter lists via the COMMAREA. If not under CICS, INFCICN must contain binary zeroes or spaces.
7. If using the reentrant ADALNKR, allocate a modified area and set its address (INFAMOD).
8. Link this program together with COR3GLI and, if not under CICS, your Adabas interface module, for example ADAUSER.
9. After calling COR3GLI, INFRC will contain 0000 and the new controls will be in effect for this session; or, INFRC will contain a non-zero return code and INFRT will contain an explanatory message.

Natural API

The Natural API is contained in library SYSCOR, subprogram CORNATI. There are also some supplied source members in SYSCOR, showing how to use the API:

- APIREV-P: example of calling CORNATI
- CORNATIA: parameter data area for calling CORNATI

To use the Natural API:

1 Ensure that subprogram CORNATI is available, by copying it to your Natural library or adding SYSCOR to your library's steplibs in Natural Security.

2 Set CORNATI-VERSION and CORNATI-FUNCTION.

3 Set the product code (CORNATI-CONTROL-PRODUCT) for which controls are to be modified (currently, this must be 'COR').

4 Set the names (CORNATI-CONTROL-NAME) of the runtime controls to be modified ('REVIEW' and/or 'REVIEW-CLIENT'), together with the required value (CORNATI-CONTROL-VALUE, 'Y' or 'N').

5 Call CORNATI.

6 After calling CORNATI, CORNATI-RC will contain 0000 and the new controls will be in effect for this session; or, CORNATI-RC will contain a non-zero return code and CORNATI-RT will contain an explanatory message.

Return Codes

These are the non-zero return codes which may be set:

Return Code	Description
0001	Invalid interface version (must be 01)
0002	Invalid function (must be CONTROLS)
0003	System Coordinator not available
0004	System Coordinator internal error
0005	System Coordinator internal error
0006	Adabas interface not linked (3GL API only)
0010	Invalid product code (must be COR)
0011	Invalid control name (must be REVIEW or REVIEW-CLIENT)
0012	Invalid control value (must be Y or N)