

Unicenter[®] CA-APAS[®] **Insight Monitor for Adabas**

Systems Guide

4.1



Computer Associates[®]

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2003 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Functional Components

Major Functions	1-1
Major Components	1-2
Data Collector – Batch Mode	1-3
Data Collector – MPM Mode	1-4
Performance History System	1-5

Chapter 2: Unicenter CA-APAS Data Collector

Data Collector and Batch Mode Usage	2-1
Purpose	2-1
Advantages	2-1
Disadvantages	2-2
Restrictions	2-2
Input Files	2-2
Sequential Command Log	2-3
Sequence of Output Records	2-3
Dual Command Log	2-3
Output Files	2-4
Execution	2-4
Sample JCL	2-4
JCL Explanation	2-5
Condition Codes at Termination	2-6
I/O Errors	2-6
Trouble Shooting	2-7
Data Collector and MPM Mode Usage	2-8
Purpose	2-8
Advantages	2-9
Potential Disadvantage	2-9
MPM Setup Changes	2-10

Input Files	2-11
Output Files	2-11
Execution	2-12
Sample JCL	2-12
JCL Explanation	2-13
User Abend Codes	2-14
I/O Errors	2-14
Pre-MPM Checkout	2-14
Trouble Shooting	2-15
Output File Considerations	2-16
Required and Optional Files	2-16
PRINT TO Files	2-16
OUTPUT-FILE Parameter	2-17
File Characteristics	2-17
Record Format	2-17
OUTPUT-EXIT Routines	2-17
COPY Files	2-17
Advantages	2-18
Modes of Operation	2-19
File Characteristics	2-19
Record Format	2-19
Sharing Output Files	2-20
Advantages	2-20
Restrictions	2-20
Sharing PRINT TO Files	2-20
Print Record Headers	2-21
Sharing OUTPUT-FILE Files	2-21
Sharing COPY Files	2-21
Disk Space Requirements	2-22
Single Disk Files	2-23
Dual Disk Files	2-23

Chapter 3: Unicenter CA-APAS User Exit Four Routine

General Considerations	3-1
Execution Without the Data Collector	3-1
Execution With the Data Collector	3-2
Usage and Processing	3-2
Basic Logic	3-3
Buffer Length Adjustment	3-3
Invocation	3-4
Adabas Command Logging Parameters	3-4
JCL Requirements	3-5
Input File	3-5
Output Files	3-5
Storage Requirements	3-5
Validation of Control Statements and JCL	3-5
Control Statement Errors	3-6

Chapter 4: Dual Output Files

Processing Overview	4-1
Specifying Dual Files	4-2
OUTPUT-FILE and PRINT TO Parameters	4-2
Print Record Header Requirement	4-3
Physical Allocation of Dual Files	4-3
File Size	4-3
File Characteristics	4-3
Formatting X / Y Files	4-3
Unicenter CA-APAS Dual File Processing Logic	4-4
Selection of Starting File (X or Y)	4-4
Copy and Reset Initiation	4-5
With Dual Copy Exit Routine	4-5
Without a Dual Copy Exit Routine	4-6
File Not Empty Condition	4-6
DUAL-COPY-EXIT Requirements	4-7
Copy and Reset Processing	4-8
DUALCPY	4-8
DUALCPY TRIM Options	4-9
DUALCPY Time Stamp Checking	4-9
PRINT TO Output File Considerations	4-10
Processing Dual Output Files	4-10
Executing DUALEXM	4-11

Executing DUALCPY	4-12
JCL Example	4-12
JCL Explanation	4-12
Control Statements	4-13
Input File Considerations	4-13
Output File Considerations	4-14
DUAL-FULL-EXIT	4-15
DUAL-FULL-EXIT OS Linkage	4-15
Return Codes	4-16
DUAL-FULL-EXIT Routine DUALOPER	4-16
Adabas Abnormal Termination	4-17
Ensuring Dual File Empty Status	4-17

Chapter 5: Local User Exit Four Routine

Usage	5-1
When Called	5-1
Processing Independent of Unicenter CA-APAS	5-1
Access To Unicenter CA-APAS Data	5-2
Command Log Control	5-2
Input Parameters	5-3
Return Parameters	5-4
Addressing Mode	5-4

Chapter 6: User-Derived Fields Exit Routine

Usage	6-1
When Called	6-1
Access To Unicenter CA-APAS Data	6-2
Input Parameters	6-2
Return Parameters	6-3
Addressing Mode	6-3
Sample COBOL Exit Routine	6-4
Exit Routine Code	6-4
Request Using Derived Fields	6-5

Chapter 7: OUTPUT-EXIT Routine

Usage	7-1
When Called	7-1
Input Parameters	7-2
Return Parameters	7-2
Addressing Mode	7-3

Chapter 8: Multiple Link Routine Exits

Adabas Link Routines	8-1
User-written Routines	8-1
Execution	8-2
Unicenter CA-APAS Driver/Stack Process	8-3
How to Use Unicenter CA-APAS Driver/Stack Process	8-4
Unicenter CA-APAS Driver/Stack Design Overview	8-5
Design Objectives	8-5
Design Summary	8-5
Driver Modules	8-7
Stack Modules	8-7
Link Routine Exit Modules	8-7
User Information Area (UIA)	8-7
UEXITB Conventions	8-8
UEXITB Driver Conventions	8-8
UEXITB Stack Routines	8-9
Stack Module Entry Format	8-10
Stack Module Entry Processing	8-10
User Information Area (UIA)	8-11
UIA Division	8-11
User Information Data Areas	8-12
User Information Work Areas	8-13
UIA Size Calculation	8-13
Link Routine Exit Modules	8-13
Receiving Control From UEXITB Driver	8-14
Returning Control to UEXITB	8-15
Addressing Modes	8-15
Adding a UEXITB Link Routine Exit Module	8-15

Accessing the UIA From Other Modules	8-16
Adabas User Exits	8-16
From UEXITA Routines	8-18
Finding the Correct User Information Data Area	8-18
Work Area for UEXITA	8-18
Adabas Link Routine Exits Reference	8-19
General Information	8-19
Register Contents at Entry to Link Routine Exit	8-19
Non-CICS	8-19
CICS	8-20
Register Contents When Returning (All Systems)	8-20
User Block (UB) Contents	8-21
Link Routine UB Fields	8-21
Adding a UIA to the User Block	8-22
UIA Usage	8-22

Chapter 9: Generating SMF Records

Purpose	9-1
Design	9-1
ADASMF Requests	9-2
Output Exit ADASMF	9-3
Step 1: Modify ADASMF	9-3
Step 2: Assemble and Link ADASMF	9-4
Step 3: Modify ADASMF	9-4

Appendix A: Output Record Header Formats

OUTPUT-FILE Record Headers	A-1
Print Record Headers	A-5

Appendix B: Structure of RSPCLASS CSECT

RSPCLASS CSECT	B-1
----------------------	-----

Appendix C: Data Collector Virtual Storage Requirements

Factors	C-1
Output Buffers	C-2
SUMMARIZE Request Table Sizes	C-3

Functional Components

This chapter describes the major functional components of Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS), relationships and data flows between components, and common options for configuration and modes of execution.

Major Functions

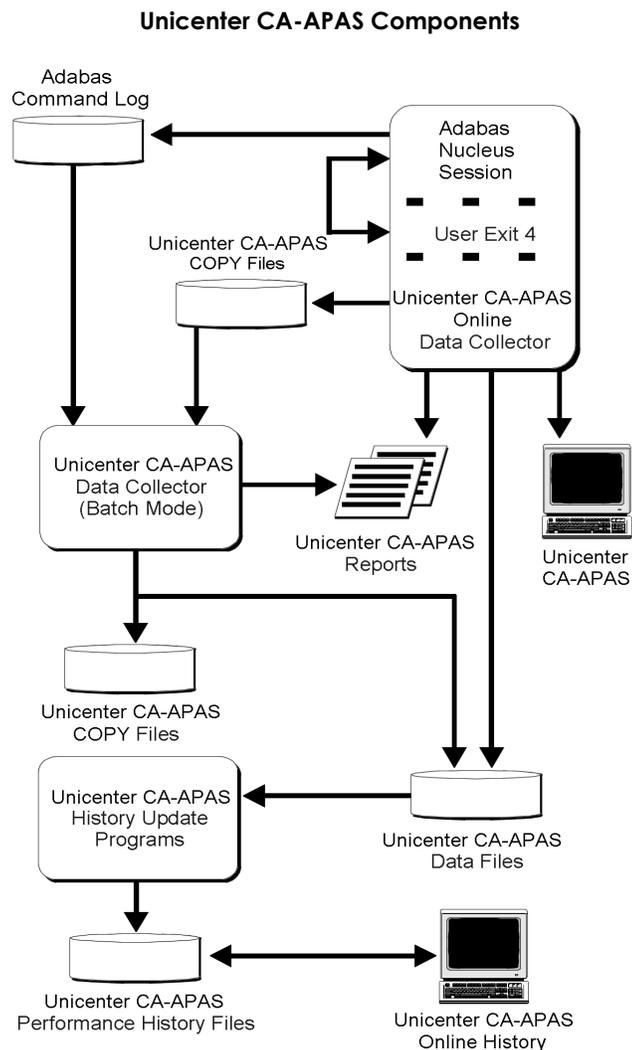
Computer Associates Adabas Performance Analysis System, Unicenter CA-APAS, performs the following major functions:

- Generates detail and summary information as hard copy reports or data files, either directly from Adabas or from files of Command Log records.
- Real-time online reports of Adabas nucleus status and activity, as well as any detail or summary reports by the Unicenter CA-APAS Data Collector.
- Selectively logs and copies command data in order to minimize the amount of command data output by the nucleus. Most analysis and summarization is done from the Data Collector without ever writing a Command Log record.
- Performance History updating and inquiry.

Major Components

Figure 1-1 shows all major Unicenter CA-APAS components and the basic data flow between them. In actual practice, several different combinations of components and modes of execution are available to suit the requirements and resources of different organizations.

Typical subsets of the components shown in the figure are broken out and described in more detail in the remainder of this chapter.



The basic flow of data through the various components of Unicenter CA-APAS is shown above; the exact configuration can be varied in a number of ways.

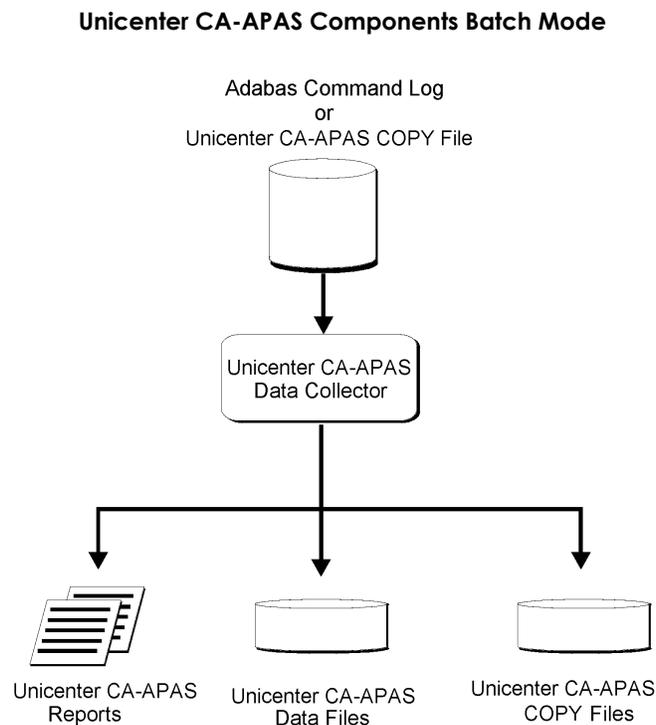
Data Collector—Batch Mode

The most elementary way to use Unicenter CA-APAS is to execute the Data Collector as a stand-alone batch job as depicted in the Unicenter CA-APAS Components graphic. This mode is immediately available as soon as the Unicenter CA-APAS libraries have been loaded from the distribution tape to disk. It requires no changes to existing Adabas processing.

When used in this mode, the Data Collector reads either an Adabas Command Log or a Unicenter CA-APAS COPY file. Either type of file may be on tape or disk. Dual (X/Y) disk files are supported.

All of the Unicenter CA-APAS command outputs except online displays may be produced in this mode. Batch mode is typically used:

- As the first step in evaluating Unicenter CA-APAS
- As the regular way of reporting command data in cases where MPM partitions are not large enough to support execution of the Data Collector with the MPM
- On an exception basis for in-depth, possibly iterative, analysis of specially selected commands such as from an application during its development



Data Collector—MPM Mode

The following graphic shows the components used when executing the Data Collector in MPM mode. This mode has significant advantages over the batch mode described previously:

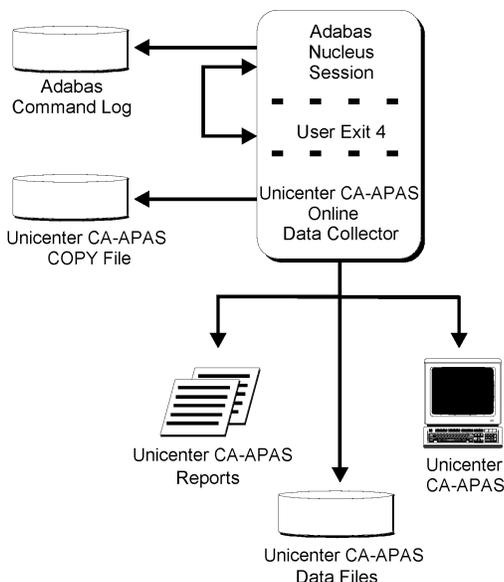
- The interactive, real-time capabilities of Unicenter CA-APAS are supported
- Writing of Command Log records by Adabas, and later reading by other programs, can be greatly reduced or entirely eliminated
- Unicenter CA-APAS printed reports, data files and COPY files become available during the MPM session with dual files or immediately after its completion

Based on user-specified criteria, the Unicenter CA-APAS User Exit 4 controls that commands, if any, Adabas writes to its Command Log. User-specified criteria in Unicenter CA-APAS COPY requests define which commands, if any, the Data Collector writes to COPY files.

If desired, the Unicenter CA-APAS User Exit 4 can call an additional exit routine provided by the using organization.

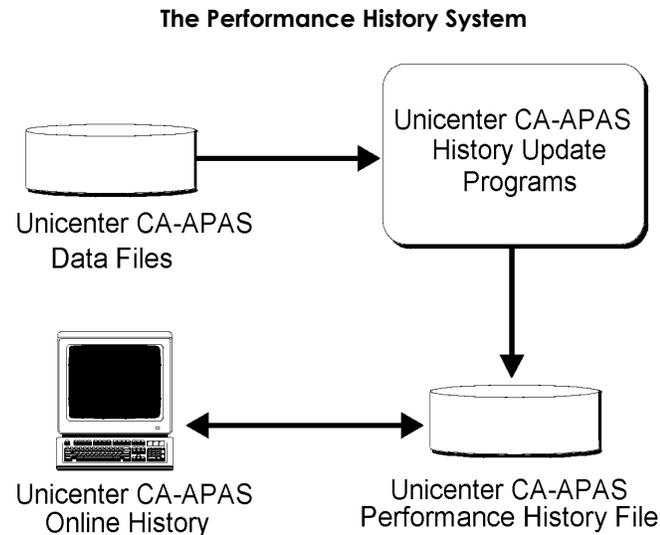
Data Collector sub-tasking and dynamic management of virtual storage minimize Unicenter CA-APAS overhead in the MPM region. Adabas throughput is normally unaffected.

Unicenter CA-APAS Components—MPM Mode



Performance History System

The figure below shows the components used with the Unicenter CA-APAS Performance History System. The Performance History System is an optional component of Unicenter CA-APAS. When used, it adds another dimension to Adabas performance analysis.



The major input to the Performance History file consists of summarized command data from Adabas nucleus sessions. These are made up of data files written by the Data Collector in either batch or MPM mode. Unicenter CA-APAS batch Natural update programs post command data for user-specified time intervals to the Performance History file.

The Online Performance History Facility provides a comprehensive set of history maintenance and inquiry functions, and is implemented as a Natural application.

Unicenter CA-APAS Data Collector

This chapter discusses operational requirements and considerations for stand-alone batch and MPM modes of executing the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) Data Collector.

Additional information about functional capabilities and usage of the Data Collector is provided in this document, as well as in the *User Guide*.

Use of the Data Collector with Unicenter CA-SpaceMan File Management for Adabas is described in the Unicenter CA-SpaceMan documentation.

Data Collector and Batch Mode Usage

In batch mode, the Data Collector obtains its input data by reading a physical Adabas Command Log file or Unicenter CA-APAS COPY file. It can then produce print-formatted reports and/or machine-readable data files as output.

Purpose

Execution of the Data Collector in batch mode serves a single basic purpose: to read a file containing Adabas Command Log records and produce printed reports and/or machine-readable data files which contain detail or summary data about Adabas commands. The types of files that may be used as input are described below.

Advantages

Batch mode execution of the Data Collector offers the following advantages over MPM mode:

- It avoids virtual storage, CPU, or I/O overhead in Adabas MPM sessions
- It can be scheduled independently from Adabas MPM sessions
- It allows multiple processing passes against a given set of records

Disadvantages

Creating an Adabas Command Log file requires both CPU and I/O overhead within the MPM session. The amount of overhead can increase Adabas response time to the extent that creation of an Adabas Command Log is unacceptable. This is the motivation for two features of Unicenter CA-APAS:

- The ability of the Data Collector to process Command Log data within a subtask of the Adabas User Exit 4
- The ability to create Unicenter CA-APAS COPY files

Both of these features are described in detail elsewhere in this document.

Restrictions

The following restrictions apply:

- Batch execution of the Data Collector does not support the use of Unicenter CA-APAS at terminals
- Neither Unicenter CA-APAS nor user-supplied Adabas User Exit 4 routines are called in batch mode
- Certain command data fields that are available to the Data Collector in MPM mode are not available in batch mode. These include:
 - IUB-LENGTH
 - IFB-LENGTH
 - ENQ-TIME
 - Instantaneous data about Adabas nucleus work pools and queues

Input Files

Unicenter CA-APAS uses two kinds of input files:

- The control card input stream is described in *Unicenter CA-APAS Writing Requests*. This is standard card-image input without sequence numbers.
ddname: DBGIN
- The file or files containing the Command Log records to be processed. In a given execution of the Data Collector, these may be any one of the following kinds of files:
 - Sequential Command Log file(s) written directly by Adabas
 - ADARES/COPY sequential files copied from Adabas dual or multiple Command Logs
 - Adabas dual or multiple Command Log disk file
 - Unicenter CA-APAS COPY file(s) written by the Data Collector
ddname: DDCLOG

Additional considerations about Adabas Command Log or Unicenter CA-APAS COPY files are discussed below.

Sequential Command Log

Management of the Adabas Command Log is documented in the *Adabas DBA Reference Manual*. Where dual logging is being used, a failure of ADARES/COPY jobs when the disk logs become full results in an incomplete Command Log.

Where Unicenter CA-APAS is being used to extract records within a user-specified time interval, the amount of Unicenter CA-APAS processing is reduced if only those tape volumes which contain records generated within that time interval are searched. A record of execution of ADARES/COPY and the volumes used is useful in this regard.

Sequence of Output Records

Output from the Data Collector is in the sequence of the input Adabas Command Log records which it reads. Detail output from EXTRACT requests is in chronological sequence only if the input Command Log files are read in chronological sequence. Simple SUMMARY requests are unaffected by the sequence of input records. Interval summary reports issue a warning message and cause an interval break when data is detected out of sequence.

Note: A generic mount of a generation data set (generations not specified) results in reading the data sets in reverse chronological order.

Dual Command Log

The Data Collector may be executed against the Adabas dual or multiple Command Log files either while Adabas is using the data sets, or at any time before the Command Log is reused by the same or another Adabas session. This mode of operation may be useful in immediate diagnosis of performance or application problems.

The Data Collector processes input records from the Adabas Command Log until a decrease in the date-time stamp is encountered. This is treated as an end of file marker. The Command Log is processed whether or not it has been copied by the ADARES utility.

Determine the active Command Log file and either process only that one, or concatenate the Command Log files in inactive-active order.

Note: Dual Command Log records are ignored by the Data Collector unless the ADARUN parameter LOGCB=YES was in effect when they were written. LOGCB is always YES when CLOGLAYOUT=5.

Output Files

For detailed information about output files, see Output File Considerations in this chapter .

Execution

In batch mode, the Data Collector is executed as a standard job step as shown in the following sample JCL.

Sample JCL

Following is a sample set of JCL statements and Unicenter CA-APAS control and request statements. An explanation of the JCL statements follows the example.

```
//JOBNAME JOB your installation's accounting info
//STEP1 EXEC PGM=APASENSU
//STEPLIB DD DSN=hlq.mlq.APASVxxx.LOAD,DISP=SHR
//* OUTPUT FILES
//DBGPRINT DD SYSOUT=A
//EXTLONG DD SYSOUT=A
//QTRHRPT DD DSN=QTRHSUM.REPORT.OUTPUT,
// UNIT=SYSDA,DISP=(NEW,CATLG),SPACE=(...)
//QTRHRFIL DD DSN=QTRHRFIL.FILE.OUTPUT,
// UNIT=SYSDA,DISP=(NEW,CATLG),SPACE=(...)
//SPECEXT DD DSN=SPECEXT.FILE.OUTPUT,DCB=BLKSIZE=4800,
// UNIT=TAPE,DISP=(NEW,CATLG)
//* INPUT FILES
//DDCLOG DD DSN=ADABAS.COMMAND.LOG.FILE,DISP=OLD
//DBGIN DD *
GLOBALS LINESIZE=132 PAGESIZE=55
CPU-ID=SYSDA DBID=1 DBNAME=PROD1;
EXTLONG: EXTRACT (LS=121) RSPCL RSP DATE TIME...
WHERE DURATION >= 1 SECS
PRINT TO EXTLONG;
QTRHSUM: SUMMARIZE COUNT PERCENT CMD-RATE...
BY FILE BY CMD
INTERVAL QTR-HOUR
PRINT TO QTRHRPT
OUTPUT-FILE QTRHRFIL;
SPECEXT: EXTRACT CMD FILE RSP FB (OF=A20)...
WHERE JOBNAME = 'PROBLEM'
OUTPUT-FILE SPECEXT;
/*
```

JCL Explanation

The following explains the sample JCL.

- `//JOBNAME JOB` is required in all cases; code according to local conventions.
- `//STEP1`, an EXEC statement, is required in all cases; it invokes the Data Collector, either directly (as in this example) or via a cataloged procedure.
- `//STEPLIB` is normally required in all cases. It identifies the load library which contains the Data Collector load modules.
- `//DBGPRINT` is required in all cases. It identifies the destination of print-formatted messages from the Data Collector.
- `//EXTLONG` is required in this example because this ddname was specified in the PRINT TO parameter of the EXTLONG request. It receives the print-formatted report from the request.
- `//QTRHRPT` is required in this example because this ddname was specified in the PRINT TO parameter of the QTRHSUM request. This report, instead of being printed directly like the EXTLONG report, is being sent to disk for later review with a data set editor and eventual printing.
- `//QTRHRFIL` is required in this example because this ddname was specified in the OUTPUT-FILE parameter of the QTRHSUM request. It specifies a disk data set to hold machine readable output. This file may later be processed by other programs.
- `//SPECEXT` is required in this example because this ddname was specified in the OUTPUT-FILE parameter of the SPECEXT request. It specifies a tape data set to hold machine readable output. This data might be used for audit or debugging purposes.
- `//DDCLOG` is required in all cases. It specifies the source of Command Log records that are to be read and processed by the Data Collector.
- `//DBGIN` is required in all cases. It identifies the source of the Unicenter CA-APAS control and request input statements.

Condition Codes at Termination

The following condition codes are returned by the program on termination:

Condition Code	Description
0	All requests were executed, and the program terminated normally.
4	All requests were executed, and the program terminated normally. However, warning messages were issued that describe unusual conditions. These messages should be examined before accepting the output as correct.
8	One or more of the requests was not able to complete normally because of an I/O error on an output data set.
12	An error was detected in a Unicenter CA-APAS control statement, and the input Command Log records were not processed.
16	An error occurred that has prevented normal execution. The error messages should be examined to determine the nature of the error.

I/O Errors

If an I/O error is detected on the input Command Log file, the program is terminated with a condition code of 16.

If an I/O error occurs on an output file, the requests using that output file are terminated, but the run continues to complete other requests. Barring any more severe errors, the program completes with a condition code of 8.

Trouble Shooting

Following are some suggestions for dealing with problems that could occur when executing the Data Collector in batch mode.

- Check the Data Collector DBGPRINT output for error or informative messages. Refer to the information given in *Messages and Codes* about particular message numbers.
- Check the Data Collector DBGPRINT output to see whether all of the intended Unicenter CA-APAS control and request statements were read and successfully syntax checked.
 - If some statements are not listed, make sure the Unicenter CA-APAS input stream is properly defined.
 - If there are syntax error messages that seem unwarranted, make sure that there were no sequence numbers in columns 73-80 of the statements.
- If time stamp sequence errors were detected, make sure that multiple files of Command Log records have been input in ascending chronological sequence.
- If a dual or multiple Command Log disk file was read but no Command Log records were read or used:
 - Verify that the dual or multiple files have been correctly defined, allocated, and formatted.
 - Verify that the file contains valid Command Log records, including the LOGCB area.
 - Verify that the file is not in a reset or empty status.

Data Collector and MPM Mode Usage

This section explains how to execute the Data Collector as an integral part of an Adabas MPM session. It discusses operational requirements and considerations applicable to this mode of execution.

Read the chapter "Unicenter CA-APAS User Exit Four Routine" in this guide before attempting to execute the Data Collector in MPM mode.

Purpose

Execution of the Data Collector as part of an Adabas MPM session allows data about Adabas processing to be captured and processed during the MPM session. Performing these functions while the MPM is active allows any of the following to be done:

- Produce the same kinds of reports, data files, and COPY files as could be produced in batch mode
- Reduce or totally eliminate the writing of Command Log records by Adabas since desired data is being captured "on-the-fly"
- Eliminate the operational burden associated with copying Adabas Command Log files from disk to tape for subsequent batch processing
- Use Unicenter CA-APAS at terminals to dynamically:
 - Change the status of Unicenter CA-APAS requests being processed by the Data Collector
 - Display detail and summary data from active Unicenter CA-APAS requests
 - Display a "snapshot" set of Adabas internal processing statistics
 - Force the Data Collector to switch between dual output files

Advantages

MPM mode execution of the Data Collector offers the following advantages over batch mode:

- Supports dynamic monitoring with Unicenter CA-APAS at any time during the MPM session
- Allows the computer processing overhead and procedural burdens associated with Adabas Command Logs to be reduced or eliminated
- Allows reports to be printed and data files to be processed during and immediately upon completion of the MPM session
- Allows for the capture of additional data items that are not available in batch mode:
 - IUB-LENGTH
 - IFB-LENGTH
 - ENQ-TIME
 - Instantaneous data about Adabas nucleus work pools and queues
- When storage of Command Log records in physical files is desired, the use of Unicenter CA-APAS COPY files in lieu of Adabas Command Log files reduce disk space requirements while minimizing the CPU and I/O overhead required to write out the data.

Potential Disadvantage

Data Collector processing causes an increase in the amount of CPU time used in the MPM session, however this does not affect Adabas response time. Since the Data Collector modules execute as lower-priority subtasks, they rarely compete with Adabas for use of the CPU. This avoids any degradation in Adabas throughput within the MPM session. However, on a CPU-bound system it may impact other jobs that are running at lower priorities than the MPM session.

MPM Setup Changes

In MPM mode, the Data Collector runs under control of the Unicenter CA-APAS User Exit Four Routine. The following changes to the normal setup for executing the MPM are required.

1. The Unicenter CA-APAS load modules must be made available for the MPM execution. This may be done either by concatenating the load library with the STEPLIB ddname in the JCL, or by copying the modules to the Adabas load library. To simplify maintenance of your load module libraries, we highly recommend keeping each vendor's modules in a separate library.

All Unicenter CA-APAS modules have been link edited with correct parameters regarding re-entrancy and reusability. Be sure that these parameters are not changed except on direction from Computer Associates. This allows the modules to be processed correctly whether their library is marked as authorized or whether they are added into a link pack area.

2. The Unicenter CA-APAS User Exit Four must be specified in the ADARUN parameters:

ADARUN UEX4=APASUEX4

3. Adabas Command Logging must be specified in the ADARUN parameters:

ADARUN LOGGING=YES	This parameter must be specified.
ADARUN LOGCB=YES	This parameter must be specified.
ADARUN LOGFB=YES	This parameter is optional and is specified to report field usage in format buffers.
ADARUN LOGIB=YES	This parameter is optional; in fact, its use is not recommended.
ADARUN LOGIO=YES	This parameter is optional and is used normally only for RABN-RANGES reporting.
ADARUN LOGRB=YES	This parameter is optional and is specified to report record buffer contents.
ADARUN LOGSB=YES	This optional parameter is used only to report the contents of search buffers.
ADARUN LOGUX=YES	This parameter is optional and does not affect the collection of ACBX data.
ADARUN LOGVB=YES	This parameter is optional and is used normally only for Natural IDs without ACBX.

4. Control statements for the Unicenter CA-APAS User Exit Four must be provided in the Unicenter CA-APAS input stream. See chapter “Unicenter CA-APAS User Exit Four Routine” for information about these statements.

The GLOBALS statement must include the parameter ENSU=YES in order for the Data Collector to be activated for the MPM session. The GLOBALS statement must also include the parameter ENSU-BUFFER=size, where “size” is the number of bytes to be allocated to a virtual storage sub-task communication buffer.

Unicenter CA-APAS User Exit Four routine copies Command Log record images into the ENSU-BUFFER. The Data Collector sub-tasks normally process the records from the buffer only during periods when the main task, Adabas, has relinquished the CPU. However, if the ENSU-BUFFER becomes full, then Adabas must wait for the sub-task to process some of the records in the buffer.

The ENSU-BUFFER should be large enough to avoid delaying Adabas. As a general guideline, the buffer should hold the Command Log record images for the longest expected sequence of consecutive commands that do not cause any Adabas physical I/O. In most situations, 32,000 bytes is adequate.

5. Any LOG, DECLARE, EXTRACT, SUMMARIZE or COPY statements follow the GLOBALS statement in the Unicenter CA-APAS input stream. These statements are described in the *Writing Requests* guide.
6. All files to be written by the Data Collector or by locally supplied exit routines must be defined.
7. The region size for the session must be increased to a value sufficient to accommodate all modules, tables, and buffers used by Unicenter CA-APAS User Exit Four and Data Collector in addition to the space needed for Adabas.

Input Files

There is one input file, the control card input stream, which is described in the *Writing Requests* guide. This is standard card-image input without sequence numbers.

ddname: DBGIN

JCL may specify in-stream (DD *) card images, a sequential data set, or a member of a partitioned data set. Concatenation may specify multiple sources.

Output Files

For detailed information about the output files, see Output File Considerations in this chapter.

Execution

In MPM mode, the Data Collector executes as a sub-task under the Unicenter CA-APAS User Exit Four. The exit is part of the main task, Adabas.

Sample JCL

Sample JCL, with Unicenter CA-APAS control and request statements, is shown below. An explanation of the JCL follows.

```
//JOBNAME JOB your installation's accounting info
//STEP1 EXEC PGM=ADARUN
//STEPLIB DD DSN=ADABAS.LOADLIB,DISP=SHR
// DD DSN=hlq.mlq.APASVxxx.LOAD,DISP=SHR
//DDPRINT DD SYSOUT=A
//DDASSOR1 DD DSN=ADABAS.ASSO,DISP=SHR
//DDDATAR1 DD DSN=ADABAS.DATA,DISP=SHR
//DDWORKR1 DD DSN=ADABAS.WORK,DISP=OLD
//DDPLOGR1 DD DSN=ADABAS.PLOG1,DISP=SHR
//DDPLOGR2 DD DSN=ADABAS.PLOG2,DISP=SHR
//DDLOG DD DSN=ADABAS.CLOG,DISP=SHR
//DDCARD DD *
ADARUN PROGRAM=ADANUC,LBP=...
ADARUN LOGGING=YES,LOGCB=YES,LOGSB=...
ADARUN UEX4=APASUEX4
/*
/* APAS OUTPUT FILES
//DBGPRINT DD SYSOUT=A
//EXTLONG DD SYSOUT=A
//QTRHRPT DD DSN=QTRHSUM.REPORT.OUTPUT,
// UNIT=SYSDA,DISP=(NEW,CATLG),SPACE=(...)
//QTRHRFIL DD DSN=QTRHRFIL.FILE.OUTPUT,
// UNIT=SYSDA,DISP=(NEW,CATLG),SPACE=(...)
//SPECEXT DD DSN=SPECEXT.FILE.OUTPUT,DCB=BLKSIZE=4800,
// UNIT=TAPE,DISP=(NEW,CATLG)
/* APAS INPUT FILE
//DBGIN DD *
GLOBALS ENSU=YES ENSU-BUFFER=32000 LOG-DEFAULT=OFF,
LINE SIZE=132 PAGE SIZE=55
CPU-ID=SYSB DBID=1 DBNAME=PROD1;
LONGCMD: EXTRACT (LS=121) RSPCL RSP DATE TIME...
WHERE DURATION > 1 SECS
PRINT TO EXTLONG;
QTRHSUM: SUMMARIZE COUNT PERCENT CMD-RATE...
BY FILE BY CMD INTERVAL QTR-HOUR
PRINT TO QTRHRPT OUTPUT-FILE QTRHRFIL;
SPECEXT: EXTRACT CMD FILE RSP FB (OF=A20)...
WHERE JOBNAME = 'PROBLEM'
OUTPUT-FILE SPECEXT;
/*
```

JCL Explanation

The following explains the sample JCL.

- `//JOBNAME JOB` is required in all cases; code according to local conventions.
- `//STEP1`, an EXEC statement, is required in all cases; it invokes ADARUN to start the MPM session.
- `//STEPLIB` is normally required in all cases. It identifies the Adabas load library concatenated to a library which contains the User Exit Four and Data Collector load modules.
- `//DDPRINT` through `//DDPLOGR2` identify Adabas data sets.
- `//DDLOG` identifies a sequential Adabas Command Log file. Either sequential or dual Adabas Command Log files must be specified in the JCL. **Do not** use DD DUMMY. If DD DUMMY is used, an ADAL05 I/O error could result and abnormally terminate the MPM.
- `//DDCARD` identifies the source of ADARUN parameters for the session. This would normally be a member of a PDS. Instream input is shown in the example to highlight ADARUN parameters which are meaningful for the Data Collector.
- `//DBGPRINT` is required in all cases. It identifies the destination of print-formatted messages from the Data Collector.
- `//EXTLONG` is required in this example because this ddname is specified in the PRINT TO parameter of the LONGCMD request. It receives the print-formatted report from the request.
- `//QTRHRPT` is required in this example because this ddname is specified in the PRINT TO parameter of the QTRHSUM request. This report, instead of being printed directly like the EXTLONG report, is being sent to disk for later review with a data set editor and eventual printing.
- `//QTRHRFIL` is required in this example because this ddname is specified in the OUTPUT-FILE parameter of the QTRHSUM request. It specifies a disk data set to hold machine readable output. This file may later be processed by other programs.
- `//SPECEXT` is required in this example because this ddname is specified in the OUTPUT-FILE parameter of the SPECEXT request. It specifies a tape data set to hold machine readable output. This data might be used for audit or debugging purposes.
- `//DBGIN` is required in all cases. It identifies the source of the control and request input statements. It may be in-stream as shown in the example or may specify a sequential data set or a member of a partitioned data set. Concatenation of multiple sources of input may be used.

User Abend Codes

If the GLOBALS keyword IGNORE-INIT-ERROR=NO is specified, the following user abend is issued if control card errors are found:

```
DBG99001E date time ERROR DETECTED IN CA-APAS COMMAND STREAM, PROCESSING  
TERMINATED
```

```
ADAM99 dbid ADABAS ABEND CODE 4000001A PSW .....
```

```
ABEND=S000 U0253
```

Explanation

A syntax error was detected in the Data Collector or User Exit Four routine control statements or requests. This abend is issued after the first user call of the session has been processed by Adabas. Unicenter CA-APAS terminates the MPM session rather than allowing the MPM session to proceed without desired information being generated.

If control card errors are found and IGNORE-INIT-ERRORS=YES is specified, processing continues and the Data Collector has a completion code of at least 12 when it terminates. Additionally, the error may cause unintended results.

I/O Errors

If an I/O error occurs on an output file, the requests using that output file are terminated, but the run continues to generate data for other requests.

Pre-MPM Checkout

If Unicenter CA-APAS detects errors in its control card input stream, it abnormally terminates the MPM to avoid a session that does not produce output you want, unless you specify IGNORE-INIT-ERRORS=YES in the Unicenter CA-APAS GLOBALS statement.

The Unicenter CA-APAS program APASENSU can be used to check control cards prior to using them with the MPM, thus avoiding aborted MPM runs. APASENSU allows the syntax of all Unicenter CA-APAS control statements and requests to be checked. This check includes verifying all referenced DD statements. Neither Adabas itself nor the Adabas Command Logs are accessed during a syntax check execution.

To perform syntax checking, use the APASENSU batch utility JCL member JCLAPASB in the Unicenter CA-APAS SOURCE library. See *Install Unicenter CA-APAS Natural Interface and Data Collector* in the chapter “Unicenter CA-APAS Installation” of the *Installation Guide*.

Trouble Shooting

The following list contains some suggestions for dealing with problems that could occur when executing the Data Collector in MPM mode.

1. Try to determine whether the Unicenter CA-APAS User Exit Four and the Data Collector were activated during the MPM session. Look for the start-up messages in the Unicenter CA-APAS DBGPRINT output or in the operator console listing. If these messages cannot be found, it may indicate that Adabas never called the Unicenter CA-APAS User Exit Four. Typical reasons might be:
 - One or more ADARUN parameters not set properly
 - The Adabas Command Log file was set to DD DUMMY
 - An operator command turned off Adabas logging before any Adabas commands were processed by the MPM session
2. Check the Unicenter CA-APAS DBGPRINT output for error or informative messages. Refer to the information given in *Messages and Codes* guide about particular message numbers.
3. Check the Unicenter CA-APAS DBGPRINT output to see whether all of the intended control and request statements were read and successfully syntax checked.
 - If some statements are not listed, make sure the Unicenter CA-APAS input stream is properly defined
 - If there are syntax error messages that seem unwarranted, make sure that there were no sequence numbers in columns 73-80 of the statements
4. If some commands processed during the MPM session appear to be missing from Unicenter CA-APAS reports, it may be due to operator commands turning LOGGING off during the session. Any commands processed while LOGGING is off are not passed by Adabas to the Unicenter CA-APAS components. Check the operator console or DDPRINT for operator commands that affect any of the logging parameters.

Output File Considerations

The following information is helpful in properly defining and handling the report and data files written by the Data Collector in either batch or MPM modes of execution.

Required and Optional Files

There are a variable number of output files:

- Standard print output, containing error messages and execution statistics
ddname: DBGPRINT
- One output file for each print request name specified in the PRINT TO parameter of EXTRACT and SUMMARIZE requests
ddnames: As specified in the PRINT TO parameters
- One output file for each name specified in an OUTPUT-FILE parameter in a COPY, EXTRACT or SUMMARIZE request
ddnames: As specified in the OUTPUT-FILE parameters

PRINT TO Files

Print-formatted reports are generated whenever a PRINT TO parameter is included in an EXTRACT or SUMMARIZE request. The format of the report is determined by the request parameters. The record size of the output report is the linesize of the report, as determined from the report format, plus one for the print control character.

All files have variable blocked formats. The record format is VBA for files created without Print Record Headers and VB for files created with Print Record Headers.

For records created without Print Record Headers, the first byte following the record descriptor word is the ANSI carriage control character.

For records created with Print Record Headers, the first byte following the Print Record Header is the ANSI carriage control character. After Print Record Headers have been removed, the resulting record has the ANSI carriage control character as the first byte following the record descriptor word.

The blocksize defaults to 4096. This may be overridden by either the BLKSIZE keyword on the DD statement or in the data set label, that is, DSCB or tape label, with the JCL value taking precedence over the data set label. The logical record length is always four (4) less than the blocksize.

OUTPUT-FILE Parameter

Machine readable output is generated whenever the OUTPUT-FILE parameter is specified in an EXTRACT or SUMMARIZE request.

File Characteristics

OUTPUT-FILES are always sequential, variable-blocked files. Blocksize defaults to 4096. This may be overridden by either the BLKSIZE keyword on the DD statement or in the data set label, that is, DSCB or tape label, with the JCL value taking precedence over the data set label. The logical record length is always four (4) less than the blocksize.

Record Format

OUTPUT-FILE format consists of a header followed by values for a variable set of fields that were specified in the request. The format of the header is shown in the appendix "Output Record Header Formats."

The default formats of the data fields are shown in *Writing Requests*. These defaults may be overridden in the request by means of the OF= parameter.

OUTPUT-EXIT Routines

OUTPUT-FILES can be written as is, or you can pre-process the output data using an OUTPUT-EXIT routine. In any EXTRACT or SUMMARIZE request you may specify the name of a locally supplied exit routine to be called before each output record is written. The specified routine may suppress writing of a given record, may modify the record before it is written, or do other kinds of processing. For specifications when writing this type of routine, see the chapter "OUTPUT-EXIT Routines."

COPY Files

The COPY file facility is provided to overcome some of the restrictions and inefficiencies that are inherent in standard Adabas Command Logging and subsequent processing of Command Log files.

A Unicenter CA-APAS COPY request causes the Data Collector to write all or selected Command Log record images to a physical file. These records can then be used as input to a batch execution of the Data Collector, which processes them as if they had come from an Adabas Command Log file or the Adabas User Exit 4. They may also be processed by any other programs which would otherwise have had to process Command Log files written by Adabas.

With the Data Collector executing in MPM mode, this facility is an alternative to having Adabas write Command Log records to Adabas sequential or dual Command Log physical files. It can be used in place of or in addition to having Adabas write physical Command Log files.

Advantages

Advantages of using the Unicenter CA-APAS COPY file facility are:

Unicenter CA-APAS
Derived Fields

Several important data fields can be obtained only when data is collected under the User Exit 4. These data fields are preserved in Unicenter CA-APAS COPY files and can be referenced in subsequent batch executions of Unicenter CA-APAS.

Fewer Physical I/Os
During MPM Sessions

Limitations on the maximum blocksize of dual or multiple Command Log files prevent the number of Command Logging writes from being optimized. Adabas dual and multiple Command Log files must be formatted by the Adabas utility ADAFRM. This is commonly done specifying the blocksize for WORK, typically in the range of 3,000 to 6,000 bytes, usually far short of the maximum blocksize possible on the device being used. Failure to use maximum blocksizes results in more physical write operations than necessary.

Maximum blocksizes possible on the device may be specified for any Unicenter CA-APAS COPY file. This results in fewer physical write operations than Adabas would have used for the same amount of data.

Sub-tasking Of I/O
Operations During
MPM Sessions

Adabas write operations to its Command Log files are not overlapped with other processing. This I/O activity can therefore degrade Adabas throughput during periods of heavy processing loads.

Writes to Unicenter CA-APAS COPY files are done from Data Collector modules executing as sub-tasks of Adabas, therefore neither the CPU time to process them nor the wait time for I/O to complete will delay Adabas. By shifting this overhead into the Unicenter CA-APAS sub-task, thereby reducing the I/O load on the Adabas task, Adabas throughput rates are increased, with resulting improvements in response times during peak periods.

Efficiency And
Convenience Of
Multiple Output Files

The COPY file facility allows Command Log records to be routed to any number of different files rather than to a single file. This subdivision into multiple files can be based on selection criteria reflecting different interests for subsequent analysis. A particular analysis process may then process (and reprocess) its own file without incurring the overhead of passing any of the other Command Log records that are not relevant. This can result in significantly shorter run times for such jobs as well as significantly reduced disk storage requirements.

By creating individual COPY files for each investigation, analysts no longer have to coordinate their use of the single, large Adabas Command Log file. One or more COPY files may be useful for exception reporting of certain categories of commands. This avoids the need to write all records to a huge Command Log file, just to extract data from a few exceptional records.

Modes of Operation

COPY files are a standard optional output from the Data Collector. No special operating environment is required. They can be created with the Data Collector executing in either batch mode or MPM mode.

COPY files may be written in single or dual file mode. For information about Unicenter CA-APAS dual file usage, see the chapter "Dual Output Files."

File Characteristics

COPY files are sequential with variable length blocked records. The blocksize can be set to optimize performance. The reduction in I/O overhead from a large blocksize should not be overlooked. The blocksize defaults to 4096. This may be overridden by either the BLKSIZE keyword on the DD statement or in the data set label, that is, DSCB or tape label, with the JCL value taking precedence over the data set label. The logical record length is always four (4) less than the blocksize.

Record Format

The record format of a COPY file is almost identical to the standard Adabas Command Log record. The record has been designed so that routines that process Adabas Command Log records are able to process COPY file records with little or no change.

The basic record, the control block portion and the variable length buffer area have the same format as the Adabas Command Log record. The Unicenter CA-APAS Derived Fields are appended to the record after the last buffer entry. The record length defined in the record descriptor word includes the Derived Fields area.

The layout of the Derived Fields is shown in files DERDEFA and DERDEFC in the Unicenter CA-APAS SOURCE library. Members RECDEFA and RECDEFC give the record layout for the Adabas Command Log record.

Sharing Output Files

Advantages

Having multiple Unicenter CA-APAS requests direct their outputs to a shared output file, PRINT TO or OUTPUT-FILE, can offer the following benefits:

- Fewer physical files to be created, specified in JCL, and post-processed
- Ability to modify currently executing requests or add new requests without disrupting an active MPM session

No one can predict from one day to the next or even from one hour to the next what new and different kinds of information about Adabas processing may be desired due to unforeseen problems or interests. Therefore, having the ability to add new requests and to alter existing requests without disrupting an MPM session is an invaluable asset.

Restrictions

The following general restrictions apply to sharing of output files by multiple requests:

- A PRINT TO file may not also be used as an OUTPUT-FILE file
- An OUTPUT-FILE may not also be used as a PRINT TO file
- An OUTPUT-FILE used for a COPY request may not also be used for an EXTRACT or SUMMARIZE request

Additional considerations about sharing output files among requests are discussed below.

Sharing PRINT TO Files

If PRINT TO parameters from multiple EXTRACT or SUMMARIZE requests specify the same ddname, then the following two factors should be considered:

- SUMMARIZE reports without INTERVAL parameters are printed one after another at the end of the run. Any number of this kind of request may share a single ddname without requiring any special consideration. This is a simple and recommended practice. It minimizes the number of ddnames that have to be defined. It requires no additional specifications in requests and no post-processing of reports.
- If SUMMARIZE reports with INTERVAL parameters and/or EXTRACT reports share a ddname with other reports, then the report lines of those reports are intermixed in the output.

Consolidation of the intermixed lines into their respective reports requires the specification of Print Record Headers for all of the requests involved, and it also requires post-processing of the shared output file before printing the reports.

Print Record Headers

Print Record Headers are requested by specifying PRH=YES in individual SUMMARIZE or EXTRACT requests, or in the GLOBALS statement. The format and contents of these headers are described in the appendix “Output Record Header Formats.”

The Data Collector checks for consistency among requests sharing a PRINT TO file. When a PRINT TO request is started, all PRINT TO requests specifying the same file are checked. If any request for that file specifies Print Record Headers, or if the GLOBALS statement specifies Print Record Headers, then records for all requests using that file must have them.

It is possible using Unicenter CA-APAS to add a new request with a PRINT TO parameter that specifies a file which has previously been opened. You should make sure that Print Record Headers were requested when the data set was opened if the new request specifies Print Record Headers. You should not start writing records with Print Record Headers to a file that already contains records without them.

Sharing OUTPUT-FILE Files

Multiple requests may specify the same file in their OUTPUT-FILE parameters. This may cause interleaving of output from different requests. Records may later be separated by a post-processor on the basis of the request name (label) contained in the header of each record. The format of the header is shown in the “Output Record Header Formats” appendix of this document.

Sharing COPY Files

Multiple COPY requests may share an OUTPUT-FILE. An individual command record is written to an OUTPUT-FILE if the WHERE clause of any COPY request writing to that OUTPUT-FILE is satisfied or if one COPY request writing to the OUTPUT-FILE has no WHERE clause. An individual command record can be written to a single OUTPUT-FILE no more than once, regardless of how many COPY requests' WHERE clauses are satisfied.

When a command record is selected to be written to an OUTPUT-FILE, the selection of the Adabas buffers, that is, format buffer, search buffer, etc., that are to be included in the COPY file record depends on the buffers requested by the individual COPY requests writing to that OUTPUT-FILE.

- A buffer is included in the COPY file record if it was requested in any of the COPY requests writing to that OUTPUT-FILE whose WHERE clause was satisfied for that record or if it was requested by a COPY request writing to that OUTPUT-FILE that did not have a WHERE clause
- If a particular buffer is not requested by any COPY request for that OUTPUT-FILE or if it is requested only by one or more COPY requests whose WHERE clauses are not satisfied, that buffer is not included in the COPY file for that command record

Note: An Adabas buffer is present in a COPY file record only if the ADARUN LOGxx parameter for that buffer has been set on so that the buffer is included with the Command Log record created by Adabas. If the ADARUN parameters specify that a buffer is not to be logged with the Command Log record, it is not present in the COPY file, regardless of the specifications in the COPY request.

Disk Space Requirements

When the Data Collector is to write reports, data files or COPY files to disk, the amount of disk space needed should be considered in advance of the execution. Two approaches are available for providing the space that is needed for each logical disk file that is to be written by the Data Collector:

- Direct the output to a single physical file that must be large enough to contain the quantity of output that is generated
- Direct the output alternately between two physical files, with each file being emptied before it is reused (dual or “X/Y” files).

Considerations for each of these approaches are discussed below.

Single Disk Files

Should an output file on disk run out of extents, the Data Collector will cease writing to that file. It also sets the completion code for the Data Collector to eight (8) if it has not already been set to a higher value. All Adabas processing and all other Unicenter CA-APAS processing continues, unaffected by this event. To prevent loss of data due to this condition, you should have some idea of the volume of data that will be generated by each request and provide file space accordingly.

In addition, each request, particularly EXTRACT requests, should include MAXRECS and MAXLINES parameters. These can prevent excessive amounts of output data from being generated by programs looping on error conditions or other unusual conditions.

Data from SUMMARIZE requests without intervals, and from EXTRACT requests reporting only exception conditions usually won't occupy a large amount of space. However, non-standard conditions, such as an application program looping on an Adabas error response code, could cause large numbers of records to be written to an exception report such as a report of non-standard Adabas response codes.

Dual Disk Files

Conventional "single" output files have inherent limitations that can lead to problems during long MPM sessions:

- The amount of data written can exceed the size of a disk file, and dedication of a tape drive is usually impractical
- The data in the file cannot or should not be accessed until the file has been closed, and this happens only when the MPM session ends or the request has been deleted via Unicenter CA-APAS

Dual output files address the size and data availability problems by allocating two physical files to one logical output file. When the first physical file becomes full, the file is closed automatically. When the output in one physical file needs to be accessed, the Unicenter CA-APAS FLIPXY command can be used to force Unicenter CA-APAS to close the file.

In either case, succeeding output goes to the other physical file of the pair. Then a job is started to copy the data in the first file to another file (tape or disk) and to reset the first file to an "empty" status so that it can again accept output when the second file becomes full. This process of "flipping" between the two physical files can continue indefinitely.

The Data Collector supports dual files in both batch and MPM modes of execution. However, use of dual files is normally practical only in MPM mode.

Commonly, each of the two physical files is made large enough to hold several hours worth of data. File size is ordinarily a critical consideration only for COPY requests that select all commands, or for EXTRACTS that select, possibly in error, all or a large number of commands.

Each file must be large enough so that the copy and reset job has plenty of time to complete before the other file becomes full. If both files become full, the operator faces the unfortunate choice of either losing data or halting the MPM until a copy and reset job can be completed. Therefore, the files' size must be selected carefully and conservatively.

Worst-case situations must be accommodated. Batch jobs can generate large numbers of Adabas commands in a relatively brief period, quickly filling a file that is too small. Another important consideration is that resources must be made available to execute the copy and reset job before the other file becomes full.

The greatest demand on dual files is often during second and third shift when batch operations, that is, large production jobs, are run. Not only do Adabas commands from batch jobs often come in large bursts, but the resources needed to run the copy and reset jobs may be tied up by long-running jobs. When this happens, a file can fill very quickly, and it may not be possible to start the copy and reset job for some time without disrupting the production work load.

In practice, most installations provide enough space in each file to allow at least four hours of operation. This usually gives plenty of time to get the copy and reset job executed without forcing a disruption of the production work load. Read the "Dual Output Files" chapter carefully before implementing Unicenter CA-APAS dual output files.

Unicenter CA-APAS User Exit Four Routine

This chapter documents the uses and functioning of the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) routine supplied for use from Adabas User Exit Four of MPM sessions.

General Considerations

The Unicenter CA-APAS User Exit Four routine can be used with or without the Data Collector in a given MPM session. The benefits available in these two modes are described below.

Execution Without the Data Collector

Sites with severe limitations on the amount of virtual storage available for use in MPM sessions may not be able to execute the Data Collector within MPM sessions. These sites can still realize several benefits from executing the Unicenter CA-APAS User Exit Four routine without the Data Collector. These benefits are:

- Derivation of special command data fields such as USER-TYPE and ENQ-TIME, with the Unicenter CA-APAS Derived Fields being appended to the Command Log records generated by Adabas for Unicenter CA-APAS COPY files.
- Improved efficiency and flexibility in the writing of physical Command Log files by Adabas:
 - With Unicenter CA-APAS, the capability to dynamically modify selection of Command Log records to be written by Adabas
 - With Unicenter CA-APAS, the capability to dynamically modify selection of buffers to be logged with Command Log records written by Adabas
 - Compression of buffers to minimum length before they are written by Adabas
- Optional invocation of a local User Exit Four routine

Execution With the Data Collector

Where sufficient virtual storage is available to support the execution of the Data Collector as part of the MPM session, the following benefits, plus all of those cited previously, are available:

- All of the Data Collector capabilities to gather, summarize, and output command data in the form of printed reports, machine-readable data files, COPY files, and online displays through Unicenter CA-APAS
- With Unicenter CA-APAS, the capability to dynamically modify requests being processed by the Data Collector
- With Unicenter CA-APAS, dynamic modification of the logical criteria that control record and buffer selection for writing by Adabas
- Optional total elimination of physical writing of Command Log files by Adabas with no loss of reporting capability

Usage and Processing

The following discussion assumes that the reader is familiar with the Command Log format and the processing by the Adabas User Exit Four as described in the *Adabas DBA Reference Manual*.

The processing logic of the Unicenter CA-APAS User Exit Four routine is described below. This helps to guide the user in achieving the results desired using the proper combination of Unicenter CA-APAS User Exit Four control cards and Adabas ADARUN Command Logging options.

The basic objective of the processing, relative to writing of Command Log records by Adabas, is to reduce the amount of data written to the Command Log file, thus reducing I/O overhead, while still allowing full information to be obtained when necessary.

Basic Logic

With appropriate Adabas Command Logging parameters in effect for an MPM session, the Unicenter CA-APAS User Exit Four routine is called by Adabas each time Adabas has finished processing a command. At each call, Adabas passes the Command Log record it has built for the completed command. The following basic steps are performed by the User Exit Four routine each time it is called:

1. If the first call, control cards are read and syntax checked.
2. Derived fields are calculated as necessary for logical selection of records or buffers.
3. If a local User Exit Four routine was specified in the control cards, it is called, just as if Adabas had called the local routine directly. The return code from the local routine can influence whether or not Adabas ultimately writes the current Command Log record to the Adabas Command Log file. For specifications for writing a local User Exit Four routine and the way its return code is used, see the chapter "Local User Exit Four Routine."
4. If the Data Collector was specified in the control cards, the record is passed to the Data Collector for processing.
5. If LOG-DEFAULT=ON, Adabas logs the basic part of the record regardless of other selection statements.
6. The record is passed by all active requests. If any LOG requests are active and the selection criteria is true for the request:
 - a. The record is logged even if LOG-DEFAULT=OFF.
 - b. The buffers specified are logged.
7. If buffers have been selected for logging, the maximum length specified in any satisfied LOG statement is used in determining the length to log.

Buffer Length Adjustment

The following adjustments are made to the lengths of buffers as they are logged:

- Format and Search buffers are only written up to the first period.
- If a record was read, the length given in the ADDITIONS-2 field is used to determine the length of the Record buffer to be written
- If a request specifies a buffer length, the minimum of the specified length or the above values is written

Invocation

The module name of the Unicenter CA-APAS User Exit Four routine is APASUEX4. It is invoked by including the module in JOBLIB or STEPLIB in the MPM JCL and by including the following ADARUN parameter in the DDCARD input to the MPM job:

```
ADARUN UEX4=APASUEX4
```

APASUEX4 is included in the distributed load library. This library may be concatenated with the Adabas load library, or the module may be moved to the Adabas load library. If the Adabas load library is authorized, the Unicenter CA-APAS load library must also be authorized.

Adabas Command Logging Parameters

LOGGING=YES and LOGCB=YES must be specified in the start-up ADARUN parameters. Any buffers that are desired for logging must be specified by their respective ADARUN parameters or via operator commands.

During the MPM session, buffer logging can be turned on and off using operator commands. The applicable commands are documented in the *Adabas Operations Manual*.

Parameter	Description
ADARUN LOGFB=YES	This parameter can be used for debugging.
ADARUN LOGIB=YES	This parameter is seldom of interest.
ADARUN LOGIO=YES	This parameter can be applied for special tuning. It is required if RABN-RANGES is specified.
ADARUN LOGRB=YES	This parameter can be used debugging and can greatly increase lengths of Command Log records with a corresponding increase in numbers of I/Os if records are being written.
ADARUN LOGSB=YES	This parameter is of major interest for performance tuning.
ADARUN LOGUX=YES	This parameter can be useful to investigate the contents of the user data generated in a link routine User Exit B.
ADARUN LOGVB=YES	This parameter may be helpful in analyzing performance of searches.

Long buffers can lead to inefficient use of blocks and frequent I/O by Adabas even when the user exit suppresses all writing of buffers. This is because Adabas bases decisions to write physical blocks on the original lengths of Command Log records rather than on the lengths after compression by the Unicenter CA-APAS User Exit Four routine. Therefore, it is recommended that Record and ISN buffer logging not be specified in ADARUN parameters or operator commands unless there is a particular need for these buffers.

JCL Requirements

The following files must be included in the Adabas MPM JCL when using the User Exit Four routine:

Input File

The control statement input stream. This is standard 80-byte card-image input without sequence numbers.

ddname: `DBGIN`

Output Files

Standard print output, containing error messages and execution statistics.

ddname: `DBGPRINT`

Output files for Data Collector requests, if the Data Collector was specified. For detailed information about these files, see to Output File Considerations in the chapter "Unicenter CA-APAS Data Collector."

Storage Requirements

If the Unicenter CA-APAS User Exit Four routine is being executed without the Data Collector, at least 100K of available virtual storage above the Adabas requirements is required. The actual working set is much smaller than this.

For information about storage requirements when the Data Collector is used, see the appendix "Data Collector Virtual Storage Requirements" in this document.

Validation of Control Statements and JCL

For instructions on validating User Exit Four routine control statements and JCL before they are used in MPM sessions, see Pre-MPM Checkout in the chapter "Unicenter CA-APAS Data Collector."

Control Statement Errors

If the Unicenter CA-APAS User Exit Four detects any errors in the control statement input, it abends the MPM with a user return code, if you specify IGNORE-INIT-ERRORS=NO in the GLOBALS statement. This occurs on receipt of the first command from Adabas and is designed to prevent conducting MPM sessions without obtaining desired data.

```
DBG99001E date time ERROR DETECTED IN CA-APAS COMMAND STREAM, PROCESSING  
TERMINATED
```

```
ADAM99 dbid ADABAS ABEND CODE 4000001A PSW .....
```

```
ABEND=S000 U0253
```

Explanation

A syntax error was detected in the Data Collector or User Exit Four routine control statements or requests. This abend is issued after the first user call of the session has been processed by Adabas. Unicenter CA-APAS terminates the MPM session rather than allowing the MPM session to proceed without desired information being generated.

If control card errors are found and IGNORE-INIT-ERRORS=YES is specified, processing continues and the Data Collector has a completion code of at least 12 when it terminates. Additionally, the error may cause unintended results.

Dual Output Files

This section describes how to establish and control Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) dual files. For an explanation of the purpose of Unicenter CA-APAS dual files and guidelines about setting the sizes of dual files, see Dual Disk Files in the chapter “Unicenter CA-APAS Data Collector.”

Processing Overview

Unicenter CA-APAS handles its dual files similar to the way Adabas handles its dual Command Logs and dual Protection Logs. However, there are some significant differences from the Adabas implementation. Some of the key points in the processing of dual files by Unicenter CA-APAS are:

- Users may specify dual files in PRINT TO or OUTPUT-FILE parameters of Unicenter CA-APAS COPY, EXTRACT or SUMMARIZE requests.
- Each logical dual file specified consists of two physical files that are referred to as the ‘X’ file and the ‘Y’ file.
- The X and Y physical files must be created and allocated before having the Data Collector first begin to process the requests that refer to the dual files.
- The physical files must be defined in the JCL for each execution of the Data Collector.
- The Data Collector decides which of the two physical files of a given dual logical output file receives data first during each execution.
- The Data Collector determines that the current physical file is full when it receives a D37 abend code from the operating system in response to a write to the file. The Data Collector handles the D37 condition without any operator intervention, and processing is not interrupted. However, the operating system issues an informative message to the operator. There is no way to suppress the D37 messages. Operators should therefore be informed in advance to ignore these messages.

- When the physical file currently receiving data becomes full, or a FLIPXY command to switch to the other file is submitted, the Data Collector closes the current file, initiates action to cause the newly-closed file to be copied and reset to an “empty” status, then opens the other physical file and begin writing subsequent data to it.
- At Data Collector normal termination, all dual files are switched, as described above, just as if a FLIPXY command had been issued for each one; this should result in all dual files being empty rather than having data left in one physical file of each pair of files.
- Adabas MPM job step JCL should be followed by conditional job steps which copies and resets each physical X or Y file in the event that the MPM terminates abnormally.

These points are discussed in more detail below.

Specifying Dual Files

Data Collector processing logic for dual files is invoked by including certain optional parameters in Unicenter CA-APAS GLOBALS, COPY, EXTRACT, and SUMMARIZE statements. The use of these optional parameters is discussed below.

OUTPUT-FILE and PRINT TO Parameters

PRINT TO and OUTPUT-FILE parameters in COPY, EXTRACT or SUMMARIZE requests may include a specification of DUAL for the logical output file. Whenever DUAL is specified, two physical files are used for the given logical file. The two physical files are called the ‘X’ and the ‘Y’ files. These files must have been created and allocated prior to the time the Data Collector uses them for the first time. The two files must be defined in the JCL for the job in which the Data Collector is executed. The following example shows how dual files are specified in Unicenter CA-APAS requests.

```
OUTPUT-FILE=(DUAL,CMDSUM)  
PRINT TO (DUAL,GENSUM)
```

The above OUTPUT-FILE parameter requires two physical files to be defined, CMDSUMX and CMDSUMY.

Output starts to one of these files. How the first one is selected is discussed later, but assume for now that the ‘X’ file is chosen. When the ‘X’ file becomes full or when a FLIPXY command is submitted through the Insight component of Unicenter CA-APAS, the ‘X’ file is closed and the ‘Y’ file is then used for the succeeding output records. Likewise, the above PRINT TO parameter requires two files, GENSUMX and GENSUMY, and they are used in the same way.

Print Record Header Requirement

When DUAL is specified with PRINT TO, a Print Record Header is automatically written as part of each output record. Unicenter CA-APAS modules use the Print Record Header to recognize that a file contains old data from a dual PRINT TO file. The format and content of the Print Record Header is described in detail in the appendix "Output Record Header Formats" of this document.

Physical Allocation of Dual Files

Each set of X and Y physical files must be created and allocated before the Data Collector is first directed to use them. Considerations and requirements for allocation of these files are discussed below.

File Size

The size of X and Y files is a critical factor for successful processing of dual files. Each physical file must be large enough to continue accepting data until sufficient time has elapsed to copy and reset the other file of a given X/Y pair. For guidelines on choosing file size, see the discussion of disk space requirements for dual disk files in Output File Considerations in the chapter "Unicenter CA-APAS Data Collector."

File Characteristics

The physical X and Y files are all sequential, variable-blocked files with a default blocksize of 4096. The blocksize may be overridden by either the BLKSIZE keyword on the DD statement or in the data set label, that is, DSCB or tape label, with the JCL value taking precedence over the data set label. The logical record length is always four (4) less than the blocksize.

Formatting X / Y Files

Pre-formatting the physical X and Y files with a Unicenter CA-APAS, Adabas, or any other operating system utility is not required.

Unicenter CA-APAS Dual File Processing Logic

An understanding of how Unicenter CA-APAS modules process dual files is essential to proper setup and smooth operations where dual files are to be used. The logic used by Unicenter CA-APAS modules at various stages of dual file processing is described below.

Selection of Starting File (X or Y)

When processing begins for a request that specifies a dual file, the Data Collector must decide which physical file, X or Y, is the first to receive data. The following logic is used:

- If both files are empty or contain records not recognizable as dual output records, the 'X' file is used first.
- If only one of the files contains records recognizable as dual output records, the other file is used first, and the copy and reset function is initiated for the file that contains the dual output records.
- If both files contain records recognizable as dual output records and the request is being added via DBGIN to a newly started Data Collector session, the open process waits for the file containing the older data to be copied and reset, then uses that file first. Adabas processing is delayed until the copy and reset job has completed.
- If both files contain records recognizable as dual output records and the request is being added dynamically via the Unicenter CA-APAS Insight component to an already active Data Collector session, the request is not started. The Data Collector issues a DBG71007E error message to the console and DBGPRINT indicating that one of the files should be emptied and the request resubmitted.

It is important to understand how dual output records are recognized by this logic. The method varies according to the types of output being sent to a given pair of dual files, as explained in the following paragraphs.

PRINT TO records for EXTRACT and SUMMARIZE requests with DUAL specified have Print Record Headers. If the first record of the first block does not have a Print Record Header, the file is assumed to be ready to accept output.

OUTPUT-FILE records for EXTRACT and SUMMARIZE requests always have an Output Record Header. This header is used by Unicenter CA-APAS to detect the presence of OUTPUT-FILE data when this type of dual file is being opened. If the first record of the first block does not have the Output Record Header, the file is assumed to be ready to accept output.

OUTPUT-FILE records for COPY requests never have any type of header. Their format conforms to that of an Adabas Command Log record with the addition of the Unicenter CA-APAS ACBX and Derived Fields at the end of the record. When a dual output file is opened for a COPY request, the format of the first record is checked to see whether it conforms to that of an Adabas Command Log record. If it does not, the file is assumed to be ready to accept output.

Copy and Reset Initiation

When the physical file, X or Y, currently receiving data becomes full or a FLIPXY command is submitted, the Data Collector closes the current file, opens the other one, and starts writing to the newly opened file. An integral part of making this switch between files is to initiate action to copy and reset the newly closed file before the newly opened file becomes full or another FLIPXY command is submitted. The Data Collector normally does this in one of the two ways explained below.

With Dual Copy Exit Routine

If the GLOBALS parameter DUAL-COPY-EXIT is specified, the named routine is invoked at the time the file switch is being made. The purpose of the exit routine is to automatically submit a job that copies and resets the indicated file. This is the preferred approach since it imposes the least burden on the operations staff. The exit routine executes as a sub-task of the Data Collector. Adabas processing continues during the copy and reset process.

The Unicenter CA-APAS module DUALEXM may be used as the DUAL-COPY-EXIT routine. DUALEXM searches a file that contains jobs that copy and reset individual dual files. DUALEXM submits the job it finds for the ddname that has just been closed. Usually, this job executes another Unicenter CA-APAS module, DUALCPY, which performs the copy and reset operation. Locally developed modules may be used instead of DUALEXM and DUALCPY, if desired. Linkage conventions for locally developed DUAL-COPY-EXIT routines are described later in this section.

The Data Collector issues messages that document its functioning for the Dual Copy Exit.

```
DBG71012I date time APAS DUAL COPY EXIT HAS BEEN ATTACHED FOR FILE ddname.
DBG71013I date time APAS DUAL COPY EXIT FOR XXXX FILE ddname HAS COMPLETED NORMALLY.
```

The above messages indicate only that the routine named in the DUAL-COPY-EXIT parameter was invoked and that it returned to the Data Collector normally. They do not imply what was done by the exit routine or whether the copy and reset operation has been done.

Having called a specified DUAL-COPY-EXIT routine as indicated by the above messages, if the Data Collector subsequently needs to start writing to the file that was to have been copied and reset, but finds this has not yet occurred, it issues messages such as the following to obtain direction from the computer operator:

```
DBG71021I date time APAS HAS WAITED 1 MINUTES FOR XXXX FILE dname TO BE UNLOADED.  
DBG71022I date time YOU MAY (A) CONTINUE TO WAIT OR (B) OVERWRITE THE FILE REPLY A OR B.
```

Note: The exact order and messages issued depends on whether the condition is encountered during initial start-up of the Data Collector or during an active Data Collector session. The examples above indicate messages issued during initial start-up.

Without a Dual Copy Exit Routine

If the DUAL-COPY-EXIT parameter is absent from the Unicenter CA-APAS GLOBALS statement, the Data Collector sends a message to the computer operator when a file switch occurs. The file that needs to be copied and reset is identified and the operator is instructed to submit a job to copy and reset that file. Data Collector processing continues without waiting for an operator response or other action.

```
DBG71031I date time XXXX FILE dname CONTAINS DATA THAT MUST BE UNLOADED.  
DBG71049I date time THE FIRST RECORD DATE AND TIME IS date time.  
DBG71041I date time FILE dname SHOULD BE UNLOADED BEFORE IT IS NEEDED AS AN OUTPUT FILE.  
DBG71040I date time NO DUAL COPY EXIT WAS PROVIDED TO UNLOAD FILE dname.  
DBG71036I date time IF YOU WANT TO START A JOB TO UNLOAD FILE dname, PLEASE DO SO NOW.
```

File Not Empty Condition

Both of the cases described above assume that the newly opened file is found by the Data Collector to be in an empty (reset) condition. In the event that the newly opened file contains dual output records that presumably have not yet been copied, the operator is given the choices shown in the following sample messages:

```
DBG71001I date time XXXX FILE dname CONTAINS DATA THAT MUST BE UNLOADED.  
DBG71002I date time WRITING TO FILE dname WITHOUT UNLOADING THE DATA WILL CAUSE  
THE DATA IN THE FILE TO BE LOST.  
DBG71023I date time YOU MAY (A) WAIT FOR IT TO BE UNLOADED, (B) OVERWRITE IT OR (C) CLOSE IT.  
REPLY A, B OR C.
```

Reply (A) should be given only if the operator expects the copy and reset job to be completed shortly. Adabas processing is held up while the Data Collector waits.

Reply (B) gives the Data Collector permission to write over the older file, thereby losing the data presently stored in it.

Reply (C) causes the Data Collector to close both the X and Y files for the logical dual output file and to discontinue writing to them for the remainder of the current execution of the Data Collector.

See the description of DUAL-FULL-EXIT to see how this selection can be automated.

DUAL-COPY-EXIT Requirements

The purpose of the exit routine is to initiate a separate copy and reset job, not to actually perform the copy and reset function itself. Having the copy and reset function run as a separate job has the following advantages:

- Copy and reset process does not slow Adabas processing
- Operations staff can control the priority of the copy and reset job
- Execution of the copy and reset job can be delayed if operational considerations dictate

The Data Collector passes the ddname of the physical file, X or Y, which needs to be copied and reset to the routine named in the DUAL-COPY-EXIT. The exit routine runs as a sub-task of Adabas.

Standard OS linkage conventions are followed. The register contents are as follows:

Register 0:	Unpredictable.
Register 1:	The address of a one-word parameter list. This contains the address of an eight-byte field that in turn contains the ddname of the file that needs to be copied and reset.
Register 13:	The address of a standard OS save area to be used by the exit routine for saving the general register contents on entry.
Register 14:	The return address that the exit routine should branch to upon its completion.
Register 15:	The entry point of the exit routine.

At termination, the exit routine should branch to the return address that was in Register 14 at entry. The contents of Register 15 is treated as the completion code of the exit routine. This does not affect the operation of Unicenter CA-APAS, but it is passed along to the DBGPRINT data set through a message.

Copy and Reset Processing

The copy and reset processing may be done by the Unicenter CA-APAS routine, DUALCPY, or by any other suitable routine selected by the user. The functioning of DUALCPY is described below; this discussion is helpful in guiding the selection or development of a different routine if desired.

If you wish merely to reset the file without copying the records, executing a standard utility such as IEBGENER with the dual output file as the output file and a null file as the input file can do this. It should be understood, however, that this results in the loss of the data in the dual output file.

DUALCPY

DUALCPY performs the following functions:

- Copies the records from an X or Y dual file to a sequential file
- Optionally, deletes the Print Record Header from PRINT TO records before writing PRINT TO records to sequential file
- Optionally, deletes the Copyright Banner box from PRINT TO reports before writing PRINT TO records to sequential file
- Optionally, compare the time stamps of successive records and treat the reading of a decreasing time stamp as a logical end-of-file
- Optionally, re-blocks the records during the copy process
- Resets the X or Y dual file to an empty status

It is important to understand that the copy and reset job must run as quickly as possible so that the dual output file is quickly able to accept new output data. DUALCPY has been written with this in mind. Because time is so critical during this job, processing of the data while it is being copied should be minimized. The data should be copied to a permanent file as quickly as possible. Any sorting or reformatting of the records should be done as part of a job or job step that runs after the dual output data set has been reset. An exception to this would be PRINT TO records where the Copyright Banner box or whose Print Record Headers are being deleted during the copy and reset process.

DUALCPY TRIM Options

The copy and reset program, DUALCPY, does not delete the Copyright Banner box or the Print Record Header during the copy process unless directed to.

Print Record Headers are deleted only if the following control statement is used:

TRIM-PRINT-HEADER = YES

The Copyright Banner box is deleted only if the following control statement is used:

TRIM-BANNER = YES

Note: The TRIM-BANNER option requires that the Print Record Header is included in the PRINT TO records.

The two TRIM options work independently of one another:

- Delete the Copyright banner box and retain the headers
- Delete the headers and retain the banner box
- Delete both the headers and banner box

DUALCPY Time Stamp Checking

All records written to dual output files have time stamps. For EXTRACT and SUMMARIZE requests, the time stamp is the record date and time stamp stored in the Print or Output Record Header. For COPY requests, the time stamp is the STCK value in the Command Log record.

Each dual output file that is receiving output should be closed when the MPM terminates. In the event of an operating system failure or an abnormal termination of the MPM, one or more of the dual output files may not be closed. The result then is that no end-of-file marker is written into the file.

The file then contains all of the new records followed by any data remaining on the storage device from previous use of the space. Since each of the records contains a time stamp, one can assume an end-of-file whenever a time stamp is encountered with a value less than the preceding record's time stamp value.

Specify the following control statement to instruct DUALCPY to compare the time stamps of successive records, and subsequently to treat the reading of a decreasing time stamp as a logical end-of-file.

TIME-STAMP-TEST=YES

If a decreasing time stamp is encountered once TIME-STAMP-TEST=YES is specified, processing continues as if DUALCPY had received an end-of-file indication. The ACPYOUT file is then closed and the ACPYIN file is reset to "empty" status.

The time stamp checking routine also performs a check of record type. If a record is read from ACPYIN that is a different type than the first record in the file, it is assumed that an end-of-file mark is missing and that this data remains from a previous use of the space. DUALCPY treats this situation in the same way it treats a decreasing time stamp.

The default value of TIME-STAMP-TEST is NO. If a value of NO is specified or if the value is allowed to default, DUALCPY continues reading the dual output file until it encounters an end-of-file mark.

PRINT TO Output File Considerations

When the Print Record Headers have been removed from a PRINT TO file, the DUALCPY output file, ACPYOUT, has a standard print format and is ready to be directed to a printer device or queue. The Print Record Header is not intended to be printed. The main purpose of the Print Record Header is to help identify and sort records prior to printing them.

The Print Record Headers contain, among other things, the request name and a sequential record number. This is provided so that PRINT TO output from several requests can be directed to a common output file, whether single or dual. The records are written in the order in which they were created. A simple sort on the PRINT TO output data set, sorting on request name and record number, before it is passed to DUALCPY to remove the headers puts the records in the order desired for printing.

Processing Dual Output Files

One way to process a dual output file with PRINT TO records from multiple requests would involve a three step process, which should be the job launched by DUALEXM.

1. Execute DUALCPY to copy the appropriate dual output file and reset it to an empty status. Do not delete the Print Record Headers in this step.
2. Pass the output records from step 1 to a sort. Sort on two fields: Request Name and Record Sequence Number.
3. Execute DUALCPY to read the sorted records and remove the Print Record Header by specifying TRIM-PRINT-HEADER=YES.

Executing DUALEXM

The DUAL-COPY-EXIT routine provided with Unicenter CA-APAS is called DUALEXM and it is distributed in the Unicenter CA-APAS LOAD library. For DUALEXM to be invoked automatically when a dual data set switch occurs, the following GLOBALS keyword must be provided:

```
DUAL-COPY-EXIT = DUALEXM
```

Two DD statements must be added to the job stream used to execute your Adabas MPM: one for APASJCL and one for APASRDR. They are used by DUALEXM to initiate the copy and reset process. The Data Collector passes the ddname of the data set that needs to be copied and reset to DUALEXM. DUALEXM searches the partitioned data set (PDS) pointed to by the APASJCL DD card for a member whose name matches the ddname of the data set to be copied and reset. It then reads the records in this member and writes them to the file pointed to by the APASRDR DD card.

The APASRDR DD card should point to the JES internal reader so that the records are accepted as a new job submitted for execution.

A sample APASRDR DD card follows. The SYSOUT class must be set according to your installation's job class conventions.

```
//APASRDR DD SYSOUT=(Z,INTRDR)
```

If a matching member name is not found in APASJCL, DUALEXM looks for a member named DEFAULT. If it finds this member, DUALEXM sends its records to APASRDR. If it does not find a member called DEFAULT, the exit routine terminates after writing the following error messages to the console:

DUALEXM*	APASJCL MEMBER <code>ddname</code> NOT FOUND. DEFAULT WILL BE USED.
DUALEXM*	APASJCL DEFAULT MEMBER NOT FOUND.
DUALEXM*	UNICENTER CA-APAS USER EXIT FAILED FOR DDNAME <code>ddname</code> .

It is important to understand that there must be one job, and hence, one member in APASJCL, for each dual output data set that could be used. There must be one job for each X data set and another for each Y data set. The job that is read in should be an execution of the program DUALCPY or a similar job which copies and resets the dual output data set.

Executing DUALCPY

DUALCPY execution requirements are discussed below. Following the JCL example and explanation of that example, further considerations that are important for DUALCPY execution are discussed.

JCL Example

The following sample JCL copies the dual output PRINT TO data set GENSUMX using DUALCPY and deletes the Print Record Header from the input records. This JCL would be inserted into the APASJCL partitioned data set as member GENSUMX for access by the DUAL-COPY-EXIT routine, DUALEXM.

```
//JOBNAME JOB YOUR INSTALLATION'S ACCOUNTING INFO
//DUALCPY EXEC PGM=DUALCPY,REGION=128K
//STEPLIB DD DSN=h1q.m1q.APASVxxx.LOAD,DISP=SHR
//ACPYIN DD DSN=APAS.GENSUMX.DATA,DISP=SHR
//ACPYOUT DD DSN=APAS.GENSUM.HISTORY(+1),
//          DISP=(NEW,CATLG)
//DBGPRINT DD SYSOUT=A
//DBGIN DD *
          TRIM-PRINT-HEADER=YES
/*
//
```

JCL Explanation

The following explains the sample JCL.

- //JOBNAME is required. It should be coded according to your installation's job accounting conventions.
- //DUALCPY, an EXEC statement, is required. The EXEC statement executes the dual output data set copy utility.
- //STEPLIB is required. It identifies the load library that contains the DUALCPY program.
- //ACPYIN is required. It identifies the X or Y dual output data set that is to be copied and reset. It is the same data set as the one identified by the GENSUMX DD card in the deck that is used to execute the Adabas MPM. It is necessary that the DISP field on both DD cards specify SHR, so that this job can access the data set while Adabas continues to run.
- //ACPYOUT is required. It identifies where the data from ACPYIN is to be written. The example shows the creation of a new member of a generation data set. For a print data set, specifying SYSOUT=A would also be valid.
- //DBGPRINT is required. It identifies the output data set where DUALCPY writes messages describing its execution.
- //DBGIN is required. It identifies the input data set where DUALCPY reads the control statements that specify how it should operate.

Control Statements

DUALCPY reads control information from ddname DBGIN, which is an 80-byte card image file. One control statement requests the deletion of the Print Record Header from PRINT TO records. Its format is:

TRIM-PRINT-HEADER=YES

Warning: Do not use this parameter when processing dual COPY files. COPY files do not have any type of headers on the records and using the TRIM-PRINT-HEADER causes the first portion of the Command Log record to be deleted.

A second control statement requests the deletion of the Copyright Banner box from PRINT TO reports. Its format is:

TRIM-BANNER=YES

Warning: Do not use this parameter when processing dual COPY files. COPY files do not have any type of Copyright banner and using the TRIM-BANNER causes the first seventeen Command Log records to be deleted.

Another control statement requests that the time stamps of consecutive records be compared. If decreasing time stamps are encountered, DUALCPY responds as if it had encountered an end-of-file mark. The format of the control statement is as follows.

TIME-STAMP-TEST=YES

Warning: Do not use this parameter when processing output files created from multiple SUMMARIZE requests. The Unicenter CA-APAS History System is such a process, as the output is written to the same file at different times. Thus, the output is out of time sequence and the use of this parameter causes loss of data.

Input File Considerations

After ACPYIN has been copied to ACPYOUT, ACPYIN is reset to an empty status. It is then ready to accept data when the next switch occurs. Although there are no requirements regarding the DSN, it is suggested that the DSN value end in X or Y, as appropriate, to match the ending character of the ddname used with the data set in the MPM JCL. DISP=SHR for this data set in both the MPM and the DUALCPY JCL is essential for proper processing of the dual file.

Output File Considerations

DUALCPY writes the records to ddname ACPYOUT. As the records are copied, they are re-blocked according to the blocksize of the ACPYOUT data set. ACPYOUT can be written to tape, disk or, for a PRINT TO data set, to a SYSOUT data set. The record format for OUTPUT-FILE data sets is VB. The record format for PRINT TO data sets is VBA. The blocksize of the ACPYOUT data set defaults to 4096.

If the copied records are to be kept in machine-readable form, generation data sets are recommended. The DUALCPY job can be set up to create a new generation at each execution. Then a subsequent job or job step can merge these records with those from previous generations after the dual output data set has been reset.

We recommend that you not use a temporary data set or a single permanent data set to store the records being copied for a dual output file. Although this works satisfactorily most of the time, problems can arise if an operating system failure occurs during the copy process. Using a single permanent data set can lead to problems if a high volume of data causes data set switching more rapidly than was anticipated. It would then be possible to have two DUALCPY jobs, one for the X data set and one for the Y data set, executing simultaneously, with both jobs trying to write to the same physical data set.

DUAL-FULL-EXIT

The condition described earlier in File Not Empty Condition in this chapter can cause MPM processing to stop while the computer operator selects what the Data Collector should do.

The DUAL-FULL-EXIT routine allows a site to automate this selection of options A, B, or C, in response to message DBG71023I.

```
DBG71001I date time XXXX FILE dname CONTAINS DATA THAT MUST BE UNLOADED.
DBG71002I date time WRITING TO FILE dname WITHOUT UNLOADING THE DATA WILL CAUSE
          THE DATA IN THE FILE TO BE LOST.
DBG71023I date time YOU MAY (A) WAIT FOR IT TO BE UNLOADED, (B) OVERWRITE IT OR (C) CLOSE IT.
          REPLY A, B OR C.
```

Note: The exit does not perform the copy and reset functions. It merely serves to specify which action the Data Collector takes in processing the “file not empty” condition.

DUAL-FULL-EXIT OS Linkage

The exit routine runs as a subroutine of the Data Collector. It uses standard OS linkage conventions, as follows:

- | | |
|--------------|--|
| Register 1: | Address of a two-word parameter list: |
| | Word 1 - Address of the eight-character ddname of the file that is to be written upon but which already contains data. |
| | Word 2 - Address of a word for storing the return code, the meaning of which is described below under “Return Codes.” |
| Register 13: | Address of an 18-word OS save area to be used by the DUAL-FULL-EXIT to save the entry registers. |
| Register 14: | Return Address. |
| Register 15: | Entry point of the DUAL-FULL-EXIT routine. |

All other register contents are unpredictable. The DUAL-FULL-EXIT must restore all register contents before returning control to Unicenter CA-APAS and it must return control to the address in Register 14 at entry to the exit.

Return Codes

The return code word is initialized to zero before the DUAL-FULL-EXIT is called. Its contents have the following effect.

- A value of 0 (zero) is equivalent to the computer operator selecting option A; that is, the Data Collector waits for the file to reset to an empty status.
- A value of eight (8) in the return code word is equivalent to the computer operator selecting option B; that is, the Data Collector resumes output to the dual files by writing over the existing data. The existing data is lost.
- A value of 16 (decimal) in the return code word is equivalent to the computer operator selecting option C; that is, the pair of dual files is closed and no further output is sent to them during this Unicenter CA-APAS session.

The DUAL-FULL-EXIT can be used in conjunction with the DUAL-COPY-EXIT to fully automate the switching, copying, and resetting of dual files.

DUAL-FULL-EXIT Routine DUALOPER

DUALOPER is the sample DUAL-FULL-EXIT routine provided with Unicenter CA-APAS and it is distributed in the Unicenter CA-APAS SOURCE library.

Determine which option (A for wait, B for overwrite, or C for close) is to be automatically initiated in response to DBG71023I messages.

Modify the DUALOPER routine accordingly, then assemble and link-edit the routine.

The following GLOBALS keyword must be provided if the DUALOPER routine is to be invoked when the condition of both dual files contain data.

DUAL-FULL-EXIT = DUALOPER

Adabas Abnormal Termination

If an Adabas MPM session terminates abnormally, without making a termination call to Unicenter CA-APAS, Unicenter CA-APAS is unable to force both of the physical files of each Unicenter CA-APAS dual file pair to an empty status at the conclusion of the Adabas MPM job step. This would leave the status of the Unicenter CA-APAS dual files unknown until the next MPM session starts, unless provision is made to handle this condition in subsequent steps of the Adabas job, as described below.

Ensuring Dual File Empty Status

To ensure that Unicenter CA-APAS dual files are forced to an empty status when the MPM job step terminates abnormally, the MPM job step should be followed by JCL for a separate job step for each X and Y physical file of each dual file pair. Each of these job steps should execute DUALCPY or an equivalent user-written program, and the EXEC statement of each job step should specify COND=ONLY. With COND=ONLY, these job steps are executed only when the earlier MPM job step has terminated abnormally.

Local User Exit Four Routine

This chapter describes the functioning and linkage conventions of a locally-written User Exit Four routine that is to be called by the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) User Exit Four routine, APASUEX4.

Usage

The UEX4 parameter of the GLOBALS control statement specifies the name of a local User Exit Four routine that is to be called by the Unicenter CA-APAS User Exit Four routine, APASUEX4. This exit allows local User Exit Four routines to function along with the Unicenter CA-APAS User Exit Four routine. The processing and linkage conventions between the Unicenter CA-APAS and local User Exit Four routines are described in the remainder of this chapter.

When Called

The local user exit routine is called after the Unicenter CA-APAS routine has developed its derived fields but before the Command Log record image has been passed to the Data Collector, if active in the current session, and before Adabas has written the Command Log record to its Command Log file.

Processing Independent of Unicenter CA-APAS

The local routine may do essentially any type of independent processing that it could do if it were the only User Exit Four routine involved. A typical example would be the generation and output of chargeback data.

Access To Unicenter CA-APAS Data

The local routine has access to fields that have been derived for the current command by the Unicenter CA-APAS User Exit Four routine as well as those fields within the Command Log record that Adabas has built. The presence of certain of the derived fields is conditional as follows:

- Certain fields are not computed unless they have been referenced by an EXTRACT or SUMMARIZE request at some point earlier in the session; these include:
 - COMMAND-COST
 - EST-CPU-TIME
 - EST-INSTRUCTIONS
- Fields acquired by the ACBX facility are absent if the Adabas call did not go through an ACBX module

The Unicenter CA-APAS Derived Field values may be incorporated into the local routine's logic. The local routine may also modify any of these values before returning. Modified values are reflected in any later command selection or reporting done by the Data Collector.

Command Log Control

When returning to Adabas, the APASUEX4 sets a return code to control whether Adabas does or does not physically write the Command Log record. The factors that govern how APASUEX4 sets this return code are identified in the decision table shown below. The local routine influences this decision process by the values it returns to APASUEX4. How the local routine provides this return code is described below in Return Parameters.

CONDITIONS	CASES										
	1	2	3	4	5	6	7	8	9	10	11
1. LOG-DEFAULT=ON	-	Y	N	N	N	N	N	N	N	N	N
2. LOG Statement(s) Present	-	-	N	Y	Y	Y	-	N	Y	Y	Y
3. WHERE Absent on Any LOG(s)	-	-	-	Y	N	N	-	-	Y	N	N
4. At Least 1 WHERE True	-	-	-	-	Y	N	-	-	-	Y	N
5. User Return Code = Zero	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Adabas WRITES CMD LOG RECORD	N	Y	N	Y	Y	N	Y	N	Y	Y	N

Note: There are several conditions that causes logging of the current record to be suppressed even if the local routine returns a zero return code.

Condition 1 depends on parameters on the Unicenter CA-APAS GLOBALS control statement.

Conditions 2, 3 and 4 depend on the use of Unicenter CA-APAS LOG statements.

Input Parameters

On entry to the local routine, registers are set as follows:

R13	Standard save area address (not back chained).
R14	Return address.
R15	Entry point address of local routine.
R1	Value at entry to APASUEX4 from Adabas.

R1 contains the address of the parameter list passed by Adabas to APASUEX4. This parameter list is described in the *Adabas DBA Reference Manual* from Software AG.

A value of zero for the Command Log record address in the parameter list indicates a final call at the end of the nucleus session. The format of the Command Log record is defined in Unicenter CA-APAS SOURCE library members RECDEFA (Assembler) and RECDEFC (COBOL).

R2	Address of the Unicenter CA-APAS Derived Fields Area.
----	---

The format of the Derived Fields Area is defined in Unicenter CA-APAS SOURCE library members DERDEFA (Assembler) and DERDEFC (COBOL). Note that the assembler or compiler adds slack bytes for necessary alignments.

R3	Contains the character string constant, "DEBUG." This is to enable the local routine to identify its caller as the Unicenter CA-APAS User Exit Four routine rather than Adabas.
----	---

R4	Address of the Adabas Intermediate User Buffer (IUB).
----	---

R5	Address of a 15-word area that contains the contents of general purpose registers 0 through 14 when APASUEX4 was entered from Adabas. Please refer to your Adabas documentation for a description of these values.
----	--

Return Parameters

The local routine provides a return code to Unicenter CA-APAS that affects whether or not the Command Log record is written by Adabas. See the earlier discussion of Command Log Control in this chapter.

This return code is placed in an action byte which is pointed to by one of the addresses in the parameter list that was provided by Adabas to APASUEX4. The format of the parameter list and the action byte are described in the *Adabas DBA Reference Manual* from Software AG.

The local routine must restore all registers, except R15, to their original contents at the time the local routine was entered from APASUEX4.

Addressing Mode

The following discussion is of particular importance to those running Adabas in 31-bit addressing mode under z/OS or OS/390.

When the local routine is called by APASUEX4, it is entered in the same addressing mode that Adabas is using. It is possible that some exit routines have to be modified, recompiled or reassembled and re-link edited to allow them to run correctly in 31-bit addressing mode. Local routines that use non-VSAM macros to perform I/O must use the 31 bit form of the macro if Adabas is running in 31 bit mode.

It is the local routine's responsibility to preserve the addressing mode under which it received control and to restore that mode before returning control to APASUEX4. Failure to do so may result in addressing exceptions and abnormal termination of APASUEX4 and the Adabas MPM.

User-Derived Fields Exit Routine

This chapter discusses the usage and linkage conventions of user-written routines to be called by the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) Data Collector before it processes Command Log record images against COPY, EXTRACT or SUMMARIZE requests.

Usage

This exit allows a local routine to calculate values for user-defined and/or to modify the values of Command Log or Unicenter CA-APAS Derived Fields. The routine called from this exit can compute values for six “user” fields based on local requirements. These fields may be reported and summarized in the same way as any other fields defined in the Unicenter CA-APAS system.

The computation of the user field values may be based on the values of any fields in either the Command Log record or the Unicenter CA-APAS Derived Fields. Lengths and formats of the six fields are described in *Unicenter CA-APAS Writing Requests*; names of the fields are:

- USERA, USERB, and USERC (alphanumeric fields)
- USER1, USER2, and USER3 (numeric fields)

The name of the routine is given in the USER-EXIT parameter of the GLOBALS statement. The processing and linkage conventions between the Data Collector and the local routine are described below.

When Called

The user routine is called after the Unicenter CA-APAS Derived Fields have been developed but before the command has been subjected to selection or processing logic of EXTRACT, SUMMARIZE, or COPY requests. The local routine is called in both batch and MPM modes of Data Collector execution.

Access To Unicenter CA-APAS Data

The local routine has access to fields that have been derived for the current command by Unicenter CA-APAS as well as those fields within the Command Log record that Adabas has built. The presence of certain of the derived fields is conditional as follows:

- Certain fields are not computed unless they have been referenced by an EXTRACT or SUMMARIZE request at some point earlier in the session; these include:
 - COMMAND-COST
 - EST-CPU-TIME
 - EST-INSTRUCTIONS
- Fields acquired by the ACBX facility are absent if the Adabas call did not go through an ACBX module.

The Unicenter CA-APAS Derived Field values may be incorporated into the local routine's logic. The local routine may also modify any of these values before returning. Modification of values is reflected in any later command selection or reporting done by the Data Collector.

Input Parameters

On entry to the exit:

R13	Standard save area (not back-chained).
R14	Return address.
R15	Entry point address of the local routine.
R1	Address of a standard parameter list that contains three addresses: Address of the Command Log record. The format of the Command Log record is defined in Unicenter CA-APAS SOURCE library members RECDEFA (Assembler) and RECDEFC (COBOL). Address of the Unicenter CA-APAS Derived Fields. The format of the Derived Field area is defined in Unicenter CA-APAS SOURCE library members DERDEFA (Assembler) and DERDEFC (COBOL). Note that the assembler or compiler adds slack bytes for necessary alignments. Address of the CQE that was built for the command being processed.

Return Parameters

The exit routine must restore all registers, except R15, to their original contents at the time the exit routine was entered from the Data Collector.

Addressing Mode

The following discussion is of particular importance to those running Adabas in 31 bit addressing mode under z/OS or OS/390.

When the exit routine is called by the Data Collector, it is entered in the same addressing mode that Adabas is using. It is possible that some exit routines must be modified, recompiled or reassembled and re-link edited to allow them to run correctly in 31 bit addressing mode. Exit routines which use non-VSAM macros to perform I/O must use the 31 bit form of the macro if Adabas is running in 31 bit mode.

It is the responsibility of the exit routine to preserve the addressing mode under which it received control and to restore that mode before returning control to the Data Collector. Failure to do so may result in addressing exceptions and abnormal termination of the Data Collector.

Sample COBOL Exit Routine

The following page contains a sample user exit routine written in COBOL. This sample is provided to show how the linkage should be established between the Data Collector and the user exit. This program would be invoked by specifying in the GLOBALS statement:

```
USER-EXIT=COBEXIT
```

Exit Routine Code

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. COBEXIT.  
REMARKS. SAMPLE COBOL USER-DERIVED FIELDS EXIT.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 TEMP-CHARGE          PIC 9(8) COMP-3.  
77 TEMP-WITH-DEC        REDEFINES TEMP-CHARGE  
                        PIC 9(3)V9(5) COMP-3.  
  
LINKAGE SECTION.  
*  
*   COMMAND LOG RECORD THAT IS ABOUT TO BE PROCESSED.  
*  
COPY RECDEFC.  
*  
*   FIELDS DERIVED BY APAS. (NOTE THAT DERIVATION OF  
*   CERTAIN FIELDS IS CONDITIONAL.)  
*  
*   THIS AREA CONTAINS THE USER FIELDS THAT MAY BE  
*   COMPUTED, REPORTED, AND SUMMARIZED BY THE USER.  
*  
COPY DERDEFC.  
*  
PROCEDURE DIVISION USING  
    COMMAND-LOG-RECORD, DERIVED-FIELDS.  
*  
* COMPUTE USER FIELDS  
*  
*   PUT A COMMAND CHARGE AMOUNT INTO USER1.  
*   (THE CHARGE FOR AN INDIVIDUAL COMMAND  
*   WILL BE VERY SMALL, SO PRECISION IS IMPORTANT.)  
*  
    COMPUTE TEMP-WITH-DEC =  
        0.002 * DER-TOTAL-IO  
        + ( 0.004 * ( DER-CPU / 1000 ) ).  
    MOVE TEMP-CHARGE TO DER-USER1.  
*  
*   GET AN ACCOUNT CODE OUT OF A TABLE  
*   SAVE IT IN THE USERC AREA  
*  
    PERFORM TBL-LOOKUP.  
    MOVE TBL-RESULT TO DER-USERC.  
*  
    GOBACK.
```

Request Using Derived Fields

The fields from the sample routine (USER1 and USERC) could be used in an Unicenter CA-APAS request such as the following:

```
CHARGIT:  SUMMARIZE
          SUM(USER1) (HD='COST', '.001', '$' PF=N7.2)
          SUM(TOTAL-IO) SUM(CPU)
          BY USERC (HD='ACCOUNT', 'CODE' PF=A12)
          PRINT TO CHARGREP;
```


OUTPUT-EXIT Routine

This chapter discusses the usage and linkage conventions of user-written routines to be called by the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) Data Collector so the user routine can process output record images that have been built for EXTRACT, SUMMARIZE, or COPY requests.

Usage

This exit allows a local routine to process output record images that have been created by the Data Collector for EXTRACT, SUMMARIZE, or COPY requests. The exit routine may change the contents of a passed record but must not change the record's length. In particular, the RDW passed by Unicenter CA-APAS must not be changed. If output records with different lengths are desired, the exit routine must construct them in its own area and write them to its own file.

The routine can suppress the writing of the record by the Data Collector. The name of the routine is given in the OUTPUT-EXIT parameter of the EXTRACT, SUMMARIZE, or COPY statement.

The processing and linkage conventions between the Data Collector and the local routine are described below.

When Called

The exit routine is called each time the Data Collector has built an output record image for the request but before the record has been written to an output file.

The routine is also called once during Data Collector normal termination processing, without a record being passed. The local routine is called in both batch and MPM modes of Data Collector execution.

Any routine named in more than one request must be coded to be re-entrant or reuseable. Failure to satisfy this requirement causes unpredictable results.

Input Parameters

On entry to the exit:

R13 Standard save area (not back-chained).

R14 Return address.

R15 Entry point address of the local routine.

R1 Address of a two-word parameter list:

Address of the output record. The contents of this word is binary zeroes on the final call.

Address of a word that can be used by the exit routine. The contents of the word whose address is stored in this part of the parameter list is preserved between calls to the exit routine. The exit routine could store into the preserved word the address of a work area, for instance. Each request has its own unique, dedicated word.

Return Parameters

A return code of 16 (decimal) in R15 suppresses writing of the record by the Data Collector. Any other value in R15 results in the record being written if an OUTPUT-FILE was specified in the request. The exit routine must restore all registers, except R15, to their original contents at the time the exit routine was entered from the Data Collector.

Addressing Mode

The following discussion is of particular importance to those running Adabas in 31 bit addressing mode under z/OS or OS/390.

When the exit routine is called by the Data Collector, it is entered in the same addressing mode that Adabas is using. It is possible that some exit routines must be modified, recompiled or reassembled and re-link edited to allow them to run correctly in 31 bit addressing mode. Exit routines that use non-VSAM macros to perform I/O must use the 31 bit form of the macro if Adabas is running in 31 bit mode.

It is the exit routine's responsibility to preserve the addressing mode under which it received control and to restore that mode before returning control to the Data Collector. Failure to do so may result in addressing exceptions and abnormal termination of the Data Collector.

Multiple Link Routine Exits

To get the most out of the material in this chapter, review the following sections:

- Adabas Link Routines to find out the purpose, benefits, and use of the Unicenter CA-APAS Driver/Stack process
- Unicenter CA-APAS Driver/Stack Design Overview to get a general idea of the included components and how they fit into Adabas operations
- UEXITB Conventions for detailed information about the architecture and functioning of the link routine UEXITB components
- Accessing the UIA From Other Modules for information on how user exits which are called in the processing of Adabas commands can use the information passed by the UEXITB exit routines
- Adabas Link Routine Exits Reference for detailed background information on the design and functioning of the Adabas link routine exit facility as provided by Software AG

Adabas Link Routines

In standard modes of operation, calls from an application program to Adabas pass through an Adabas link routine. The Adabas link routine executes in the same address space as the application program which called it. The Adabas link routine then passes information about the application call on over to Adabas which is executing in its own separate address space.

User-written Routines

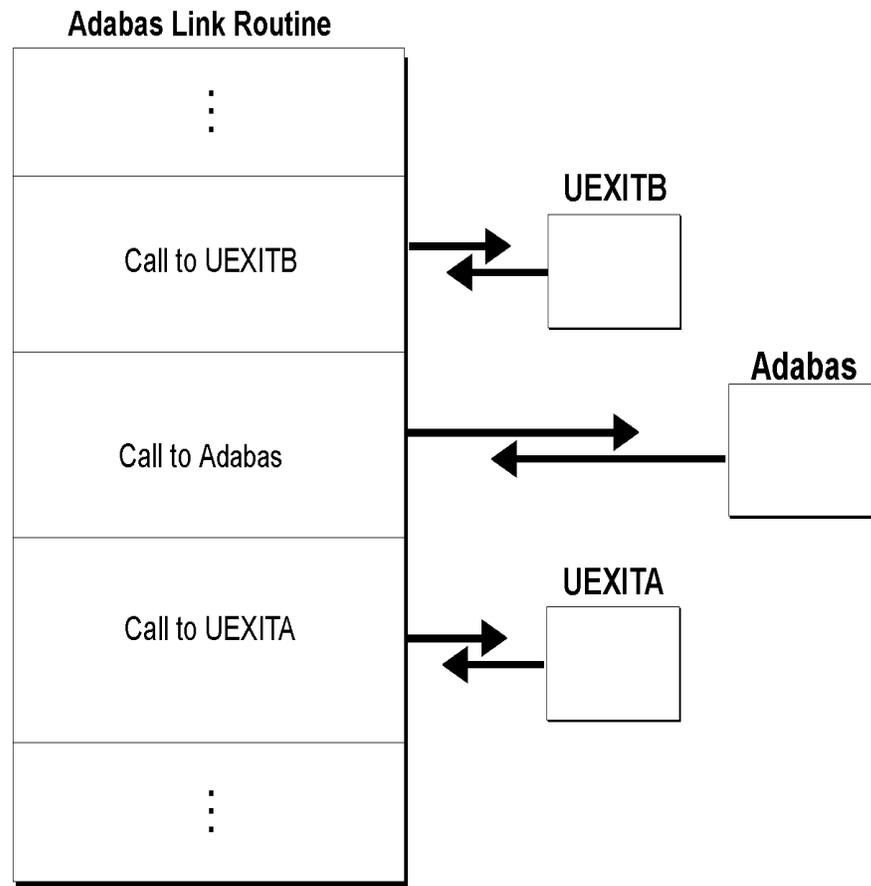
Execution of user- or vendor-written code along with the Adabas link routines is a common requirement. Such routines typically might:

- Test security authorizations and block the call from being processed any further when appropriate
- Modify some of the calling parameters
- Develop additional information to pass along to user exit routines within Adabas and/or to code within the Adabas link routine that is executed after return from Adabas

Execution

Adabas provides two standard user exit points in its link routines. One exit, UEXITB, is called immediately Before the link routine transfers control to Adabas. The other exit, UEXITA, is called immediately After Adabas returns control to the link routine. The flow of control between these modules is shown here:

Control Flow between Modules



Further, the Adabas link routines provide a storage area which is made available to both of the link routine exits and to all of the user exits in Adabas itself. This storage area, the User Information Area, can be used to pass data between these various exit routines at their respective stages in the processing of a call to Adabas.

Unicenter CA-APAS Driver/Stack Process

A Problem	<p>The standard Adabas link routine exits provide for only one User Information Area and for calling only a single routine from each of the standard exit points. However, it is often desirable to have more than one user- or vendor-written routine executed in conjunction with the Adabas link routines and for each of these routines to have its own dedicated storage area for passing information.</p> <p>The user organization may have one or more independent routines to capture and to pass along information for accounting, statistical or audit purposes. In addition, security systems or other types of systems from vendors may need to execute routines along with the Adabas link routine.</p> <p>Since the standard Adabas link routine itself does not provide for multiple independent exit routines to be called and to share the User Information Area, some additional logic is required.</p>
A Solution	<p>To help meet the need described above, the Unicenter CA-APAS Driver/Stack process is provided. The Unicenter CA-APAS Driver/Stack process provides a design and supporting software that allow any number of independent routines to share an Adabas link routine exit point and share the User Information Area.</p> <p>The Unicenter CA-APAS Driver/Stack process is already in use at several hundred Adabas sites in connection with Unicenter CA-APAS. Other vendors whose products need to share the Adabas link routine exits with Unicenter CA-APAS and user-written routines have also adopted it.</p>
Benefits	<p>Using Unicenter CA-APAS Driver/Stack process offers your organization the following benefits:</p> <ul style="list-style-type: none">■ Avoid the time and expense of developing an in-house approach■ Easily add, change and delete exit routines independently from one another■ Guaranteed compatibility with exit routines from other users or vendors who adopt the Unicenter CA-APAS Driver/Stack approach
Items Provided	<p>The Unicenter CA-APAS Driver/Stack process consists of this chapter and the modules included in the Unicenter CA-APAS SOURCE library. This chapter provides a description of the Driver/Stack process and explains how you implement it. The Unicenter CA-APAS SOURCE library contains supporting code that you modify to meet your specific requirements. Code for the following operating systems and teleprocessing monitors is provided:</p> <ul style="list-style-type: none">■ IBM z/OS and OS/390: batch, TSO, CICS, COM-LETE, IMS■ IBM VSE: batch, CICS, COM-LETE■ Fujitsu MSP: batch, AIM/DC■ Hitachi VOS3: batch, DCCM/XDM

How to Use Unicenter CA-APAS Driver/Stack Process

The basic steps in using the Unicenter CA-APAS Driver/Stack process are shown in the following table.

Step	Action
1	Read these sections in this manual: <ul style="list-style-type: none">■ Unicenter CA-APAS Driver/Stack Design Overview■ UEXITB Conventions■ Accessing the UIA From Other Modules
Follow steps 2-7 for each Adabas link routine with which you want to use one or more exit routines.	
2	Select the appropriate Driver and Stack routines from among those that are provided in the Unicenter CA-APAS SOURCE library.
3	Write new exit routines or modify the existing routines to conform to the Unicenter CA-APAS Driver/Stack conventions. Assemble and link edit each of these routines.
4	Add an entry to the Stack routine for each exit routine this Driver is to call. Assemble and link edit the Stack routine.
5	Compute the size the User Information Area that is required for this Adabas link routine.
6	Use the value computed in Step 5 to modify the Adabas link routine. Assemble and link dit the Adabas link routine.
7	Link edit a working copy of the Adabas link routine into a test library and include the following modules: <ul style="list-style-type: none">■ Adabas link routine from Step 6■ Driver module from Step 2■ Stack module from Step 4■ Each of the exit routines from Step 3

Unicenter CA-APAS Driver/Stack Design Overview

This section describes the objective of the link routine exit modules, the components included, and how they fit into Adabas operations.

Design Objectives

The purpose of Unicenter CA-APAS Driver/Stack process is to let you have one or more independent functions accomplished at a given Adabas link routine exit point. Each of these functions is accomplished in a separate link routine exit module that is not provided as a standard part of Adabas.

Our Unicenter CA-APAS Driver/Stack approach allows an unlimited number of link routine exit modules to use a single Adabas link routine exit without having to communicate with each other. This allows you to add and to delete link routine exit modules without having to resolve inter-dependencies.

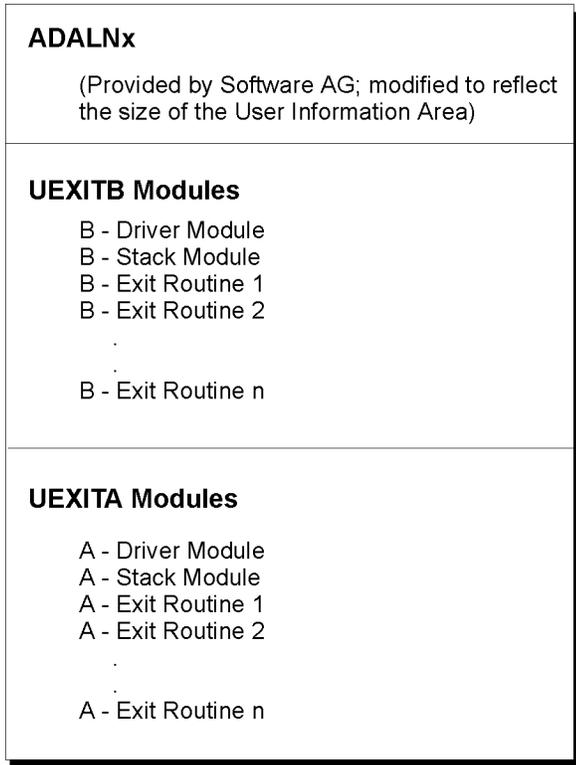
Design Summary

The Unicenter CA-APAS Driver/Stack process uses three types of modules to accomplish the design objectives described above:

- One Driver module for each standard exit that is to be used
- One Stack module for each Driver module
- One or more link routine exit modules that accomplish the functions you require at link routine exit time

These three types of modules are described here. This figure shows their organization relative to an Adabas link routine.

Module Organization and Adabas Link Routine



Another key element in the Unicenter CA-APAS Driver/Stack design is the way the User Information Area (UIA) is shared by the link routine exits. This also is described below.

Driver Modules

The Driver module for either of the Adabas link routine standard exits is an interface between the Adabas link routine and the user- or vendor-written link routine exits that perform the functions you require. The Driver receives control from the Adabas link routine, then calls each of the link routine exit modules coded in the Stack module, one after another.

The accompanying Unicenter CA-APAS Driver/Stack routines in the source library consists of only UEXITB Drivers, and most of the discussion of Drivers in this document focuses on UEXITB Drivers. This is because UEXITB is more commonly used than UEXITA and a UEXITB Driver must do a few things that are not required of a UEXITA Driver. If you need a UEXITA Driver, you should be able to clone it from one of the UEXITB Drivers. UEXITB Drivers are described in more detail in UEXITB Conventions in this chapter.

Stack Modules

The Stack module provides information about the link routine exit modules to its associated Driver module. The Stack module contains a table entry for each link routine exit that the Driver module must call. The table entry gives the entry-point address, character string identifier, and storage requirements of the link routine exit. You must revise the Stack module each time you add or delete a link exit routine. Stack modules are described in more detail in UEXITB Conventions in this chapter.

Link Routine Exit Modules

These are the routines, written by users or vendors, which perform the desired functions each time the Adabas link routine is executed. By conforming to Unicenter CA-APAS Driver/Stack conventions, these routines may be added, changed, and deleted without ever impacting one another. Detailed specifications for these routines are given in the UEXITB Conventions in this chapter.

User Information Area (UIA)

The Stack modules specify how much storage each link routine exit module needs for passing data and how much storage it needs for transient Work Area. The UEXITB Driver module uses this information from the Stack module to subdivide and format the User Information Area (UIA) in such a way that each link routine exit module's storage requirements are satisfied.

Organization and use of the UIA is described in more detail in the UEXITB Conventions in this chapter.

UEXITB Conventions

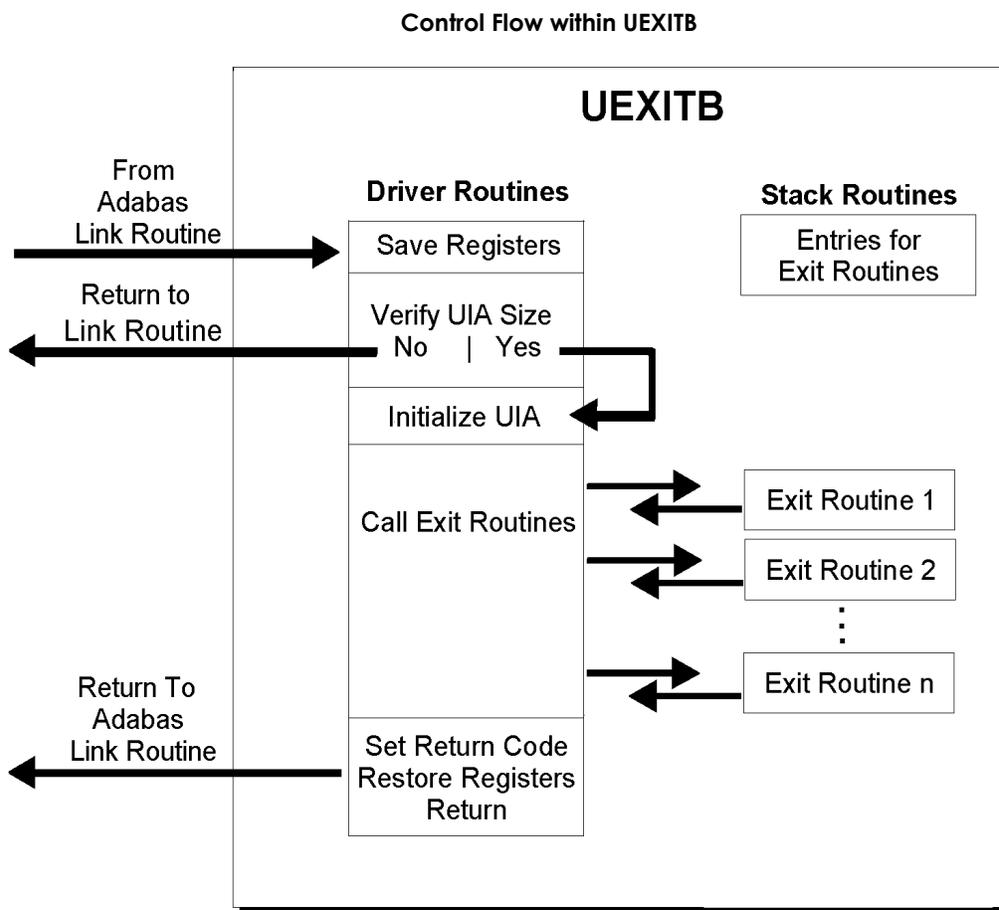
This section provides detailed information about the architecture and functioning of the UEXITB components.

UEXITB Driver Conventions

A UEXITB Driver routine is required for each operating system and teleprocessing monitor. Each UEXITB Driver performs the following tasks:

- Receives control from the Adabas link routine before control is transferred to Adabas.
- Saves register contents at entry.
- Adds up the storage requirements found in the Stack entries and verifies that the amount of User Information Area (UIA) specified in the Adabas link routine is sufficient. If the allocated amount of UIA is not large enough to satisfy all of the link routine exit module storage requirements, the Driver routine returns control to the Adabas link routine without calling any of the exit modules. If the UIA is large enough, then the Driver routine continues as outlined below.
- Formats the UIA so that it can be shared by the various link routine exit modules and Adabas user exits.
- Calls each of the link routine exit modules, in the sequence they are listed in the Stack routine, preserving the largest return code.
- Stores the largest return code into the ACBRSP field of the Adabas Control Block (ACB).
- Restores registers, putting the return code in Register 15, and returns control to the Adabas link routine.

The figure shown below diagrams the control flow within UEXITB when the Driver and associated modules are used.



Each UEXITB Driver routine must conform to the conventions of the environment in which it executes; for example, CICS, COM-LETE, etc. Additional information about UEXITB Driver functioning for special cases is given in this section in connection with descriptions of other UEXITB components.

UEXITB Stack Routines

The UEXITB Driver gets the information it needs about each link routine exit module from its associated Stack routine. The Stack routine contains a series of table entries, one entry for each link routine exit module that the Driver is to call. You add or delete UEXITB link routine exit modules by adding or deleting their Stack module entries.

Stack Module Entry Format

Each Stack module has a four-byte count of the number of entries it contains. That count is followed by the specified number of 12-byte entries, one for each link routine exit module. Each 12-byte entry is formatted as follows:

Displacement	Contents
0	V-type address constant containing the entry point of the link routine exit module.
4	Half-word length of the User Information Data Area for this link routine exit module.
6	Half-word length of Work Area required for this link routine exit module.
8	Four-byte Unique Character Identifier that is used to identify the User Information Data Area.

Stack Module Entry Processing

The UEXITB Driver determines from all of the 12-byte entries in its Stack routine how large the UIA must be, and it then formats the UIA accordingly.

Once the UIA has been formatted, the UEXITB Driver calls each of the routines pointed to by the Stack module entries.

If the address for a routine has not been resolved, that is, is zero or negative, the UEXITB Driver skips the call, but the routine's User Information Data Area and Work Area are still allocated.

The reason for allocating a User Information Data Area even when no link routine exit module is present is to allow a way to allocate space in the UIA that is filled in by an Adabas user exit and then passed to a subsequent Adabas user exit or to a UEXITA link routine exit module. The reason for allocating a Work Area for a non-existent routine is to provide work space needed in UEXITA routines.

If you need such areas, create an entry in the Stack module with a zero entry point, but with a non-zero size of the User Information Data Area and/or Work Area, and a non-blank four (4) character identifier.

The Adabas user exits and the UEXITA routine are then able to find and use the areas which have been allocated by the UEXITB Driver using the four character identifier.

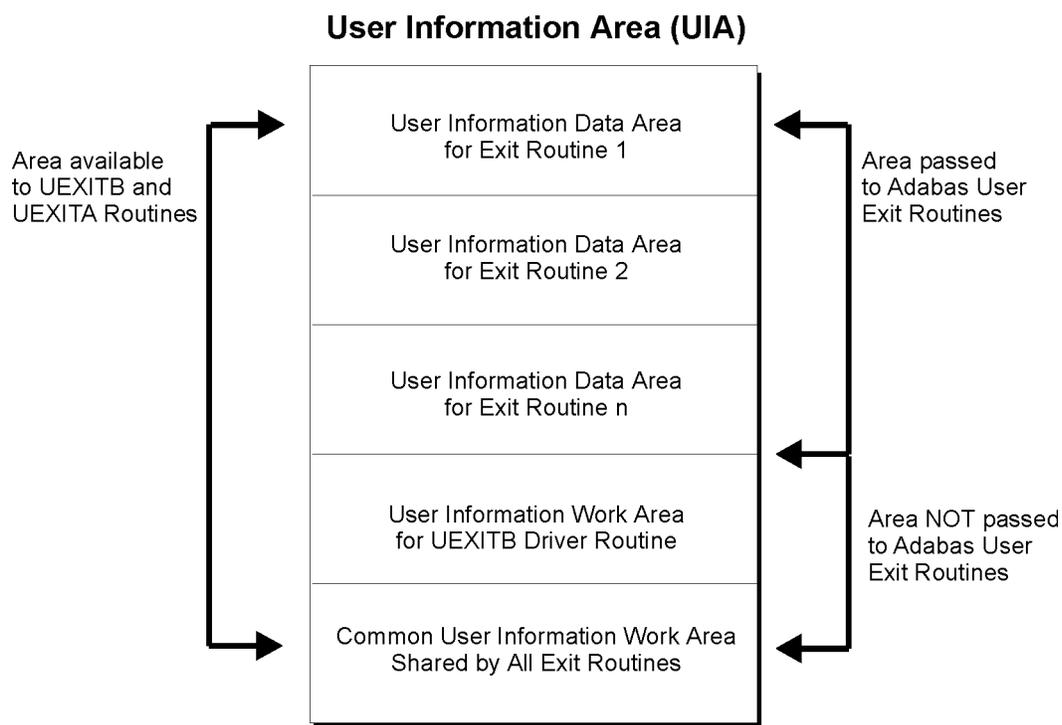
User Information Area (UIA)

The UEXITB Driver module divides the UIA that is supplied by the Adabas link routine into two segments.

- A segment consisting of one or more User Information Data Areas, one for each link routine exit module that requires one.
- A segment consisting of a single User Information Work Area that can be used by any of the UEXITB or UEXITA link routine exit modules.

UIA Division

The User Information Data Areas are subsequently accessible by Adabas user exit routines; the User Information Work Area is not. This division is illustrated below:



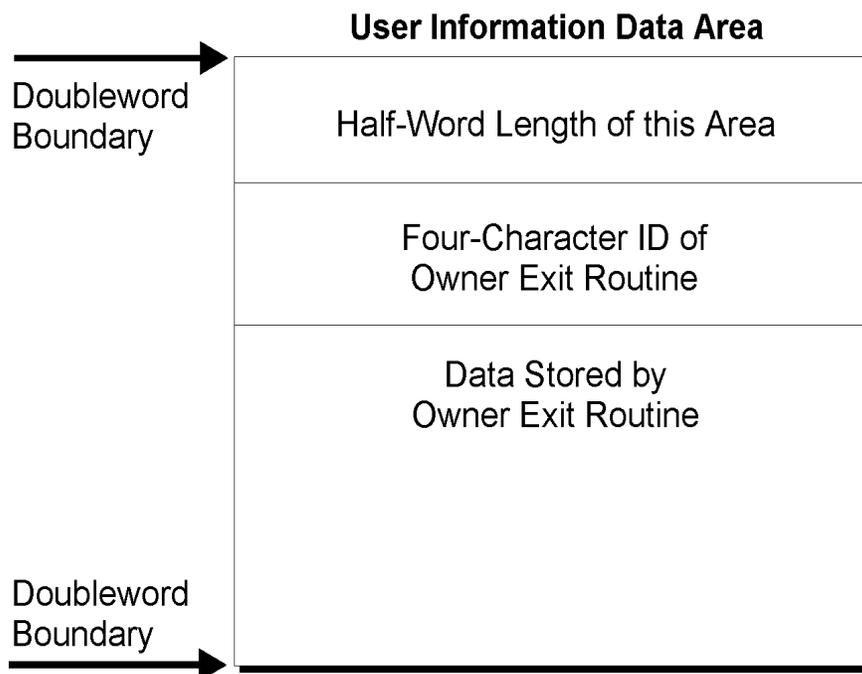
User Information Data Areas and User Information Work Areas are described in more detail below.

User Information Data Areas

If a link routine exit module is to pass data to a subsequent exit, it must store that data in its User Information Data Area.

The size of its User Information Data Area is specified in its entry in the UEXITB Stack module. The UEXITB Driver routine allocates and initializes the User Information Data Area within the User Information Area that is provided by the Adabas link routine.

Each User Information Data Area is allocated to start and end on a double word boundary. Each data area is headed by a two-byte inclusive length of the data area. This length is followed by a four (4) character identifier that identifies the link routine exit module which “owns” this data area. The figure below shows this organization of the User Information Data Area.



Each link routine exit module stores the data that it wants to make available to subsequent link routine exits or to one or more Adabas user exits following the six-byte header.

User Information Work Areas

This type of area differs from a User Information Data Area in that it cannot be used to pass information, and it may be used by other link routine exit modules that are called by the Driver.

The Work Area is allocated to start and end on a double word boundary. The Work Area is available for temporary working storage that is required by the link routine exit module, for example, to provide a save area for any subroutines which it calls.

Having this Work Area available within the UIA makes it unnecessary for a link routine exit module to use a supervisor routine to allocate working storage. This should make the exit run faster.

Only one Work Area is allocated by the UEXITB Driver. Its size is equal to that of the largest one specified in the UEXITB Stack module. Each link routine exit module may use the area as it is called in series by the UEXITB Driver.

UIA Size Calculation

The size of the UIA must be inserted into the Adabas link routine before it is assembled. This size is computed by adding the following values:

- The size of each link routine exit module's User Information Data Area, where the size of each area has been rounded up to a double word boundary
- The size of the largest Work Area needed by any of the link routine exit modules, rounded up to a double word boundary
- 120₁₀ bytes, which is the size of the Work Area required by the UEXITB Driver itself

Link Routine Exit Modules

Each exit routine is called in turn by the UEXITB Driver, and each can perform its functions without being concerned about other exit routines.

Each exit routine must preserve initial conditions so that any subsequent exit routines get correct input data.

Avoid interdependencies between exit routines; this maintains the ability to add or to delete a link routine exit module without having to change the logic in any other routine. The Unicenter CA-APAS Driver/Stack conventions accomplish this when used with the User Information Area format described above.

Receiving Control From UEXITB Driver

The UEXITB Driver calls each of the link routine exit modules in series. The entry conditions for each link routine exit module are as follows:

Registers 0 and 2 through 12

Have the same values that they had on entry to the UEXITB Driver.

Register 1 Points to the following parameter list..

Register 0 Address of the User Block (UB) that was built by the Adabas link routine.

Register 4 Address of the User Information Data Area to be used by this link routine exit module.

Register 8 Address of the User Information Work Area to be used by this link routine exit module.

Register 13 Has the address of a standard 18-word save area to be used by this link routine exit module to save the register contents at entry.

Register 14 Has the return address.

Register 15 Has the entry point of this link routine exit module.

Note: Under CICS, Register 2 points to a 16-word save area that will have been used by the UEXITB Driver to store the register contents when it was called by the Adabas link routine. This save area must not be used by your exit routine. Use the save area pointed to by Register 13 to save the registers at entry to your exit routine.

The User Information Data Area contains its inclusive length in the halfword at displacement 0 from the start of the User Information Data Area and a four (4) character identifier that is unique to this link routine exit module at displacement of two (2). The remainder of the area, starting at a displacement of six (6), has been set to zeroes and can be modified by the link routine exit module.

The contents of the Work Area is unpredictable and can be modified by the link routine exit module.

Returning Control to UEXITB

Before returning to the UEXITB Driver:

- Registers 0 through 14 must be restored to the values they contained at entry to the link routine exit module
- Register 15 must be set according to the description below in Register Contents When Returning (All Systems) for the appropriate system
- The link routine exit module should branch to the address that was in Register 14 at entry to the routine

Addressing Modes

The following discussion is important to sites running the Adabas link routines in 31 bit addressing mode under z/OS or OS/390.

When a link routine exit module is called by the UEXITB Driver, it is entered in the same addressing mode that the Adabas link routine is using. It is possible that some link routine exit modules must be modified, recompiled or reassembled and re-link edited to allow them to run correctly in 31 bit addressing mode. Those routines using non-VSAM macros to perform I/O must be modified to use the 31 bit form of the macro if the link routine is running in 31 bit mode.

It is the link routine exit module's responsibility to preserve the addressing mode under which it received control and to restore that mode before returning control to the UEXITB Driver. If it fails to do so, this could cause addressing exceptions and abnormal termination of UEXITB and the Adabas link routine.

Adding a UEXITB Link Routine Exit Module

What follows describes how to add a UEXITB link routine exit module to those already present. Be sure to follow the standards described previously when adding a UEXITB link routine exit module.

Step	Action
1	Determine the sizes of the User Information Data Area and Work Area required by the new routine. Zero size is acceptable if either area is not needed.
2	Examine the existing versions of the Stack module and make the following changes. <ol style="list-style-type: none"> a) Insert a 12-byte entry for the new routine, specifying the routine's entry point name, User Information Data Area and Work Area lengths, and unique four-character identifier. b) Reassemble and link edit the modified Stack module.

Step	Action
3	Adjust the value of LNUINFO in the Adabas link routine source to include space for the new User Information Data Area.
4	Increase the value of LNUINFO if the new size of the Work Area is greater than the size of the Work Area specified for any of the other link routine exit modules. Do not forget to allow 120 ₁₀ additional bytes of Work Area space for the UEXITB Driver. CICS users must also verify that the value of USERSAV is at least 72.
5	Reassemble the Adabas link routine.
6	Link edit the Adabas link routine together with the UEXITB Driver, the Stack module, and all of the link routine exit modules.

Accessing the UIA From Other Modules

This section describes how user exits that are called in the processing of Adabas commands can use the information passed by the UEXITB link routines.

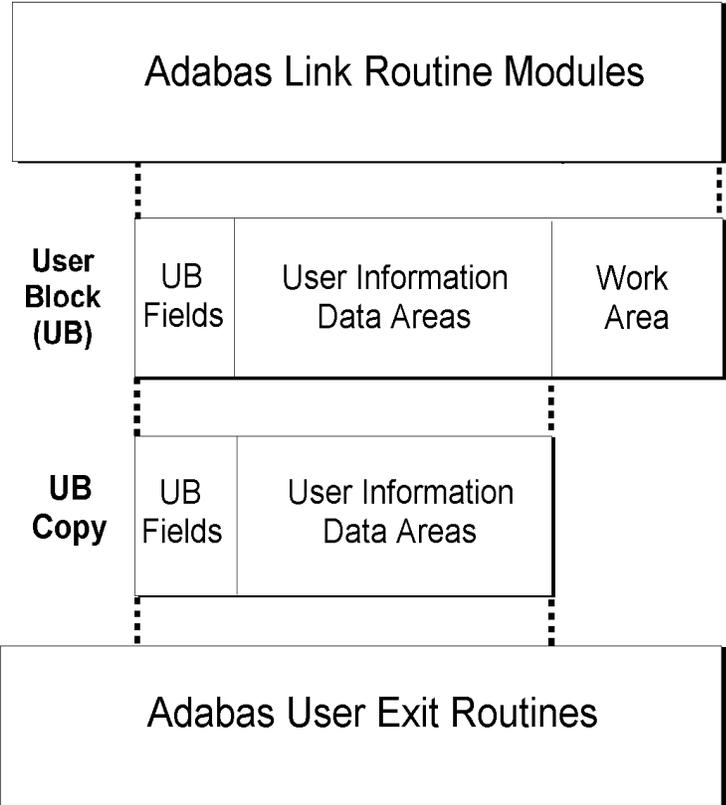
Adabas User Exits

Adabas user exits are able to find the User Information Data Area they are looking for using the following information.

- On entry to the user exit routine, Register 1 contains the address of a 4-word parameter list.
- The fourth word in the parameter list contains the address of the Adabas Command Queue Element (CQE) for the current Adabas command.
- Label CQEAUI in the CQE refers to a copy of the User Information Area that was passed to the Adabas MPM.
- The address in CQEAUI is the address of UBLUINFO, the half-word length of the UIA. This length value is “non-inclusive” so it does not include the two bytes occupied by UBLUINFO itself.
- The UIA immediately follows UBLUINFO.
- The first User Information Data Area starts at the beginning of the UIA.
- Each User Information Data Area starts with its inclusive length followed by the four-character identifier that was taken from the Stack module.
- The Adabas user exit routine can step through the UIA looking for the User Information Data Area(s) that it needs.

As shown below, the Work Areas are not included in the copy of the UB that is made available to Adabas user exit routines.

User Block (UB) Copy



From UEXITA Routines

UEXITA link routine exit modules have access to the entire UB, including the entire UIA, not just the portion that was passed to the Adabas MPM based on the value of UBLUINFO.

Finding the Correct User Information Data Area

UEXITA and UEXITA exit routines are able to find the User Information Data Area they are looking for using the following information.

- On entry to the routine, Register 1 contains the address of the User Block (UB)
- At X'3E' in the UB, the half-word labeled UBLUINFO contains the length of the UIA
- The UIA immediately follows the UBLUINFO half-word, starting at X'40' of the UB
- The first User Information Data Area starts at the beginning of the User Information Area
- Each User Information Data Area starts with its inclusive length followed by the four-character identifier that was taken from the Stack module
- The Adabas user exit routine can step through the UIA looking for the User Information Data Area(s) that it needs

Work Area for UEXITA

The UEXITA routine is able to use the Work Area that was allocated by the UEXITB Driver in the UIA. This Work Area starts immediately after the last User Information Data Area. The displacement of the start of the Work Area from the start of the UIA is the value of UBLUINFO.

When coding a UEXITA link routine exit module, be sure that the Work Area allocated by the UEXITB Driver is big enough to meet the needs of the UEXITA routine. One way to do this is to create an entry in the Stack module that has no UEXITB link routine exit module entry point, a zero User Information Data Area length, a blank value for the four-character identifier but the actual Work Area size required by the UEXITA routine.

Adabas Link Routine Exits Reference

This section describes selected details about the link routine exit facility as found in Adabas. The Driver routines interface directly with the facility described here. Using the Unicenter CA-APAS Driver/Stack approach and writing your own link routine exits to be called by a Driver routine is easier if you understand this information.

General Information

You can add either or both of two user exit routines to any Adabas link routine. You add the exits by link editing them into the Adabas link routine.

The user exit routines must be named according to their functions. UEXITB receives control **B**efore a command is passed to Adabas. UEXITA receives control **A**fter Adabas has processed a command and control has been returned to the Adabas link routine.

Register Contents at Entry to Link Routine Exit

Non-CICS

On entry to either link routine exit on systems other than CICS, the general register contents is as follows:

R1	Address of the User Block (UB). This is the parameter list that the link routine builds and then passes to Adabas. It contains a complete description of the Adabas command.
R13	Address of an 18-word save area used by the link routine exit for saving register contents at entry.
R14	Return address.
R15	Entry point of the link routine exit.

CICS

On entry to either link routine user exit under CICS, the general register contents is as follows:

R1	Address of the User Block (UB). This is the parameter list that the link routine builds and then passes to Adabas. It contains a complete description of the Adabas command.
R2	Address of the 16-work save area at the end of the UB that was allocated by LNKOLSC and can be used by the link routine exit to save the contents of the register at entry.
R12	Address of the CICS DFHEIBLK.
R13	Address of the CICS DFHEISTG.
R14	Return address.
R15	Entry point of the link routine exit.

Register Contents When Returning (All Systems)

On return from the link routine exit, the values in all registers except Register 15 must be restored to the values they were at entry.

The value in Register 15 on return from UEXITA does not affect execution of the link routine. However, the value in Register 15 on return from UEXITB is very important. If the value of Register 15 is set to anything other than zero when control is returned to the link routine from UEXITB,

- The command is not sent to Adabas
- Control is returned to the program that called the link routine

The UEXITB routine should set the value of the ACBRSP field to a non-zero value to indicate to the program that called the link routine that it has suppressed execution of the command. Response code 216₁₀ is reserved for this purpose.

User Block (UB) Contents

There are several flags and storage areas in the UB to support the link routine exits and to make the UIA available to the Adabas user exits.

We do not give a complete description of the UB here. Rather, we describe only those fields which are related to the link routine exits.

Link Routine UB Fields

The fields listed below are User Block fields related to the link routine exits.

UB Field	Description
UBFINUB	<p>The X'40' bit at +X'10' of the UB. This bit is used to support NET-WORK and ADA-NET, where a command might be passed between the link routines so that it is processed by the appropriate MPM. The User Information Area should be modified only by the UEXITB for the first link routine to receive the command. UBFINUB is the flag that allows you to determine this.</p> <p>If this bit is zero, this UEXITB should modify the User Information Area.</p> <p>If UBINFOUB is one, an earlier UEXITB has already been called for this command and the User Information Area should not be modified further.</p>
UBLUNIFO	<p>A half-word at +X'3E' of the UB. The link routine that calls UEXITB with UBFINUB set to zero inserts the length of the User Information Area, which starts at UBUNIFO. This value can be decreased by UEXITB if less than the entire User Information Area is to be made available to the Adabas user exits and to UEXITA.</p> <p>Note: The value of UBLUNIFO must never be increased because the link routine will abnormally terminate.</p>

UB Field	Description
UBINFO	The User Information Area, which starts at +X'40' of the UB, is the area used to pass data between the link routine exits and the Adabas user exits. It is allocated by the link routine before UEXITB is called. Its length is determined based on values set within the Adabas link routine, as is determined based on values set within the Adabas link routine, as described below. If the length of the the area is greater than zero, the area's first two bytes are set to zero by the link module before UEXITB is called.

Adding a UIA to the User Block

If you desire a UIA in the UB, you must modify, assemble, and then link-edit the Adabas link routine source code together with all of the modules which make up UEXITB and/or UEXITA.

UIA Size	The label LNUINFO in the Adabas link routine source is for an EQU instruction that defines the size of the UIA. Set the value to the size you require for the UIA. After assembling the Adabas link routine, verify that the half-word with the label LUINFO correctly picks up this length.
CICS	CICS users must make one additional change to their Adabas link routine; the label USERSAV is an EQU instruction that defines the size of a register save area that is passed to the link routine exit by UEXITB or UEXITA. This value must be set to at least 72 ₁₀ .

UIA Usage

The UIA can be used to pass data from the first link routine exit (UEXITB) to any of the Adabas user exits or to UEXITA. Any of the Adabas user exits can use this area to pass information to subsequent user exits or to UEXITA.

Each Adabas command receives its own UIA. The UIA cannot be used to pass information between, or to accumulate data across, multiple Adabas commands. Any of the other Adabas user exits can do this, but they must do it in storage which they have allocated to themselves. It is the Adabas user exit's responsibility to preserve the address of such an area between calls.

Adabas moves the UIA into storage which is addressable by Adabas and its user exits. Label CQEAUI in the CQE refers to a copy of the UIA that is passed from the link routine exit. CQEAUI contains the address of UBLUINFO. This is a halfword that contains the length of the UIA and immediately precedes it.

Generating SMF Records

This chapter discusses using Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) to collect Adabas usage and performance information and create SMF records.

Purpose

Many data centers require that daily or weekly reports be generated to review system usage or application performance using SMF data and want to include Adabas usage in these reports. Others want to include Adabas usage in an existing SMF charge back and billing system. Another possible use of Adabas related SMF data is for auditing purposes. While Adabas does not provide a means for generating SMF records, you can use Unicenter CA-APAS to produce Adabas related SMF records.

Design

Using Unicenter CA-APAS requests to collect the Adabas performance and usage information, a Unicenter CA-APAS OUTPUT-EXIT is then used to format and write SMF records directly to the SMF data sets. Each record written to SMF by the SMF OUTPUT-EXIT contains a standard SMF record header with subtypes followed by the data collected by the Unicenter CA-APAS requests.

The SMF OUTPUT-EXIT suppresses the writing of the output records by the Data Collector to Unicenter CA-APAS-allocated data sets. Therefore, no additional data sets are required to be allocated or defined in the MPM when using Unicenter CA-APAS to generate Adabas SMF records.

The following SMF related members are in the Unicenter CA-APAS SOURCE library:

ADASMFR	Ccontains two Unicenter CA-APAS SMF SUMMARIZE requests.
ADASMFY	Contains the Unicenter CA-APAS SMF OUTPUT-EXIT.
ADASMFJ	Contains a DSECT layout of the data produced by the Unicenter CA-APAS requests found in ADASMFR.
ASMSMFY	A JCL job stream to assembly and link edit the ADASMFY exit.

ADASMF Requests

The Unicenter CA-APAS requests used to collect the Adabas performance and usage information are in the Unicenter CA-APAS SOURCE library member ADASMFR. ADASMFR contains of two SUMMARIZE requests – one for batch and TSO users, and a second for on-line TP users. These requests are a sample of the type of Adabas performance data and usage information that can be written to SMF; they can be modified to meet your sites SMF requirements.

The ADASMFR requests include ACBX and Derived fields such as ENQ-DUR, USER-TYPE, and USER-ID, which requires the ACBX facility be implemented for all environments accessing the database before these fields contain meaningful data. For instructions on installing the ACBX facility, see Part 3: Install the Unicenter CA-APAS ACBX Facility in the chapter “Unicenter CA-APAS Installation” in the *Unicenter CA-APAS Installation Guide*.

Note: If you modify the ADASMFR requests and the amount of data collected to be included in the SMF records exceeds 226 bytes, the ADASMFY OUTPUT-EXIT must be modified to define a larger SMFRECRD DSECT.

Output Exit ADASMF

The OUTPUT-EXIT routine used to create and write the SMF records is located in the Unicenter CA-APAS SOURCE library member ADASMF. The exit is called by the Unicenter CA-APAS requests at the end of the INTERVAL specified in the requests.

The ADASMF exit contains several user specified options used to build the SMF header fields:

- SMF System Id
- SMF Subsystem Id
- SMF Subtype
- SMF Record types

See the comments in ADASMF for an explanation of options available for each SMF header field.

Step 1: Modify ADASMF

The OUTPUT-EXIT ADASMF in the Unicenter CA-APAS SOURCE library contains several user specified options used to build the SMF header.

Edit ADASMF. Review and modify the following user options:

1. &SYSID – SMF System Id.
2. &SUBSYS – SMF Subsystem Id.
3. &SUBTYP – SMF Subtype.
4. &RECTYB – Batch and TSO Record Type.
5. &RECTYTP – TP Record Type.

Step 2: Assemble and Link ADASMF

Use the following steps to modify the ASMSMF JCL member in the Unicenter CA-APAS SOURCE library to assemble and link edit ADASMF.

1. Use the AIEDIT macro to change the values of the environmental variables in the ASMSMF member in the Unicenter CA-APAS SOURCE library.
2. Submit the ASMSMF job to assemble and link edit ADASMF. If you receive a non-zero return code, correct the JCL and resubmit the job.

Note: The ADASMF load module must be linked into a Load library, such as the Unicenter CA-APAS LOAD library, that is concatenated in the MPM STEPLIB DD statement.

Step 3: Modify ADASMF

Edit the ADASMF member in the Unicenter CA-APAS SOURCE library. This member contains two SUMMARIZE requests – one for batch and TSO users, and a second for on-line TP users. These requests are a sample of the type of Adabas performance data and usage information that can be written to SMF; they can be modified to meet your sites SMF requirements.

Add the two ADASMF requests to DBGIN control card and input stream for each MPM that you want to produce Adabas usage SMF records for.

Note: If you modify the ADASMF requests and the amount of data collected to be included in the SMF records exceeds 226 bytes, the ADASMF OUTPUT-EXIT must be modified to define a larger SMFRECRD DSECT.

Output Record Header Formats

This appendix documents the format and contents of the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) record headers for PRINT TO and OUTPUT-FILE records. The purposes and uses of these headers are discussed in *Writing Requests* and in other parts of this document.

OUTPUT-FILE Record Headers

Every OUTPUT-FILE record written by the Data Collector contains an OUTPUT-FILE header. The format is shown below. You should examine member OUTHDRA to obtain the latest version of this DSECT.

DSECT for OUTPUT-FILE Record Header

```

OUTRECRD DSECT
* THIS DSECT MAPS THE START OF AN OUTPUT FILE RECORD.
OUTRECDW DS      XL4      RECORD DESCRIPTOR WORD.
OUTEXNAM DS      CL8      EXTRACT NAME.
OUTDBNAM DS      CL8      DATABASE NAME.
OUTDBID  DS      XL2      DATABASE ID.
OUTCPUID DS      CL8      CPU ID.
OUTRDATE DS      CL8      REPORT DATE.
OUTRTIME DS      CL6      REPORT TIME.
* INTERVAL SUMMARY DESCRIPTORS
OUTINSTD DS      CL8      INTERVAL START DATE.
OUTINSTT DS      CL6      INTERVAL START TIME.
OUTINDAY DS      ZL1      DAY OF WEEK (1-7) .
OUTINWKY DS      ZL2      WEEK OF YEAR (0-53) .
OUTINEND DS      CL8      INTERVAL END DATE.
OUTINENT DS      CL6      INTERVAL END TIME.
OUTFRSDT DS      CL8      1ST RECORD IN INTERVAL; DATE.
OUTFRSTI DS      CL6      1ST RECORD IN INTERVAL; TIME.
OUTLSTDY DS      CL8      LAST RECORD IN INTERVAL; DATE.
OUTLSTTI DS      CL6      LAST RECORD IN INTERVAL; TIME.
          DS          X      RESERVED.
OUTBRKLV DS      XL2      BREAK LEVEL.
* DETAIL RECORD DESCRIPTORS
          ORG OUTINSTD
OUTRECDT DS      CL8      RECORD DATE FOR DETAIL RECORD.
OUTRECTI DS      CL6      RECORD TIME FOR DETAIL RECORD.
OUTRECDY DS      ZL1      DAY OF WEEK (1-7) .
OUTRECWK DS      ZL2      WEEK OF YEAR (0-53) .
* SUMMARY DESCRIPTORS
          ORG OUTINSTD
OUTSMSTD DS      CL8      SUMMARY START DATE.
OUTSMSTT DS      CL6      SUMMARY START TIME.

```

```

OUTSMDAY DS      ZL1      DAY OF WEEK(1-7) .
OUTSMWKY DS      ZL2      WEEK OF YEAR(0-53) .
OUTSMEND DS      CL8      SUMMARY END DATE .
OUTSMENT DS      CL6      SUMMARY END TIME .
      ORG
OUTRECTY DS      CL2      RECORD TYPE .
* IS = INTERVAL SUMMARY
* GT = GRAND TOTAL OF INTERVAL SUMMARY
* SU = SUMMARY WITHOUT INTERVALS
* DE = DETAIL RECORD
OUTHDRLN EQU *-OUTRECRD HEADER LENGTH .
OUTDATA DS       0X      START OF DATA .
    
```

Following is a brief explanation of the fields in this header:

Field	Description
OUTRECDW	Standard variable length record header. COBOL and other high-level language programs do not see this word.
OUTEXNAM	Name coded as the label for the request.
OUTDBNAM	Eight-character DBNAME coded as a GLOBALS parameter.
OUTDBID	Two-byte binary database-id taken from the Adabas nucleus.
OUTCPUID	Eight-character CPU-ID coded as a GLOBALS parameter.
OUTRDATE OUTRTIME	Date and time of the Data Collector run which created this file.

The following fields apply to interval summaries.

Field	Description
OUTINSTD OUTINSTT	Date and time the interval started.
OUTINDAY	Day of the week the interval started, with Sunday being day 1.
OUTINWKY	Week of the year the interval started, with the first week being week 0.
OUTINEND OUTINENT	Date and time the interval ended.
OUTFRSDT OUTFRSTI	Date and time of the first record in the interval (zero if no records in interval).
OUTLSTDY OUTLSTTI	Date and time of the last record in the interval (zero if no records in interval).

Field	Description
OUTBRKLV	Break level for the record being written. This ranges from 1 for the highest level subtotal record, to n, where n is the number of control-break fields and the most detailed level of break. The grand total record for the interval has a break level of 0. If SUBTOT=OFF is specified, all have a break level of n. Fields 1-4 are zero if no Command Log records are processed for the request. An example of how this could occur is an MPM session during which no Adabas commands are processed.

The following fields redefine the header at OUTINSTD for detail records:

Field	Description
OUTRECDT OUTRECTI	Detail record date and time.
OUTRECDY	Day of the week, with Sunday being day 1.
OUTRECWK	Week of the year, with the first week being week 0.

The following fields redefine the header at OUTINSTD for summaries with no interval:

Field	Description
OUTSMSTD OUTSMSTT	Date and time of first record read for the summary.
OUTSMDAY	Day of the week, with Sunday being day 1.
OUTSMWKY	Week of the year, with the first week of the year being week 0.

The following fields are present in all records. The first two, however, are not meaningful for detail (EXTRACT request) records.

Field	Description
OUTSMEND OUTSMENT	Date and time of the last record read for the summary.
OUTRECTY	Type of the record. This is a two-byte alpha field with the following values: IS interval summary record. GT the grand total summary record written at end of run for INTERVAL request. SU table record for SUMMARIZE request with no INTERVAL. DE detail record (EXTRACT request).

Print Record Headers

Record headers are optional for records written to PRINT TO files. Where used, the Print Record Headers have the format shown below. You should examine member PRTHEADR to obtain the latest version of this DSECT.

DSECT for Print Record Header

```

PRTHEADR DSECT
PHDREQNM DS    CL8          REQUEST NAME.
PHDRECNO DS    XL4          RECORD NUMBER, BINARY.
PHDDBNAM DS    CL8          DATABASE NAME.
PHDDBID  DS    XL2          DATABASE ID.
PHDCPUID DS    CL8          CPU ID.
PHDREPDT DS    CL8          REPORT DATE.
PHDREPTI DS    CL6          REPORT TIME.
PHDRECDT DS    CL8          RECORD DATE.
PHDRECTI DS    CL6          RECORD TIME.
PHDRECDY DS    PL1          DAY OF THE WEEK; Sunday=1.
PHDRECWK DS    PL2          WEEK OF THE YEAR; 0-53.
          DS    XL1          RESERVED.
PHDRECTY DS    CL2          RECORD TYPE.
* IS = INTERVAL SUMMARY
* GT = GRAND TOTAL OF INTERVAL SUMMARY
* SU = SUMMARY WITHOUT INTERVALS
* DE = DETAIL RECORD
PHDLNGTH EQU  *-PRTHEADR    HEADER LENGTH.
PHDFIRST DS    X            START OF PRINT DATA.

```

Following is a brief explanation of the fields in the header. PRINT TO records are written in variable blocked format, so allowance should be made for the block and record descriptor words that precedes the header fields shown.

Field	Description
PHDREQNM	The name coded as the label for the request that caused this record to be generated.
PHDRECNO	The number of this record in binary. Records are numbered consecutively within each request.
PHDDBNAM	The eight-character database name provided as a GLOBALS parameter.
PHDDBID	The two byte binary database-id taken from the Adabas nucleus.
PHDCPUID	The eight-character CPU-ID provided as a GLOBALS parameter.
PHDREPDT	The date of this report in year-month-day format.
PHDREPTI	The time of this report in hour-minute-second format.
PHDRECDT	The date of this record in year-month-day format.
PHDRECTI	The time of this record in hour-minute-second format.

Field	Description
PHDRECDY	Day of the week, with Sunday having a value of 1.
PHDRECWK	Week of the year, starting with Sunday. The first week of the year is week 0.
PHDRECTY	A two character record type, with the following values: IS – interval summary. GT – grand total of interval summary. SU – summary without intervals. DE – detail record from an EXTRACT request.
PHDFIRST	The start of the print record; the carriage control character.

Structure of RSPCLASS CSECT

This appendix describes the Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) definition of classes into which Adabas response codes have been grouped for reporting purposes. The information provided here is intended to show how any response code could be changed to a different class. A partial listing of the RSPCLASS CSECT source code is provided here for your convenience in becoming familiar with its physical and logical structure.

The source code for the entire module is included in the source library. The comments in the source code explain the structure. Dump the RSPCLASS CSECT within the APASINTM load module to verify current actual contents.

Writing Requests contains detailed discussions of each class and specific response codes included in some of the classes.

RSPCLASS CSECT

The following is a partial listing of the RSPCLASS CSECT.

```
RSPCLASS TITLE 'RSPCLASS -- RESPONSE CODE CLASSES'
***** @V33004
*(C) 2003 COMPUTER ASSOCIATES INTERNATIONAL, INC. * @V41001
***** @V33004
* RESPONSE CODE CLASS DEFINITIONS: ***** ADABAS V7.4 SM-01 @V41048
*
* DEFAULT ASSIGNMENTS AS OF 11 NOV 2002 @V41048
*
* THIS CSECT CONSISTS OF A 256-BYTE TRANSLATE TABLE WITH EACH BYTE
* ASSOCIATED POSITIONALLY WITH ONE OF THE ADABAS RESPONSE CODES 0-255
* AS INDICATED BELOW. EACH BYTE CONTAINS THE VALUE OF ONE OF THE
* RESPONSE CODE CLASSES DEFINED BELOW, THUS ESTABLISHING THE CLASS TO
* WHICH ITS ASSOCIATED RESPONSE CODE BELONGS.
*
* THE PURPOSE OF THESE CLASS DEFINITIONS IS TO ALLOW CONVENIENT
* SPECIFICATION OF EXCEPTION REPORTING BY RESPONSE CLASS INSTEAD OF
* BY INDIVIDUAL RESPONSE CODES. THE CLASS NUMBERS AND MEANINGS HAVE
* BEEN ORGANIZED SO THAT INCREASING CLASS NUMBERS REFLECT INCREASING
* SEVERITY OF ACTUAL OR POTENTIAL PROBLEMS. A FEW RESPONSE CODES
* HAVE MULTIPLE POSSIBLE MEANINGS AND COULD LOGICALLY FALL INTO
* MULTIPLE CLASSES; EACH OF THESE RESPONSE CODES HAS BEEN ASSIGNED TO
* THE HIGHEST SEVERITY CLASS INTO WHICH IT MIGHT FALL.
*
```

* TO CHANGE THE CLASS TO WHICH A RESPONSE CODE IS ASSIGNED,
 * ZAP THE NEW CLASS VALUE INTO THE TRANSLATE TABLE BYTE ASSOCIATED
 * WITH THE RESPONSE CODE. THE ZAP SHOULD BE APPLIED TO THE "RSPCLASS"
 * CSECT WITHIN THE LOAD MODULE "APASINTM" IN THE APAS LOAD LIBRARY.

CLASS NO.	MEANING
0	ALL OTHER (NORMAL, NON-ERROR RESPONSES)
1	USER LOGIC, DATA OR SPECIFICATION ERROR
2	USER ERROR IN FORMAT, RECORD, SEARCH OR VALUE BUFFER
3	CONTENTION FOR RECORD HOLD
4	TIMEOUT, OR CHECKPOINT REQUEST DENIED
5	CONFLICT BETWEEN SESSIONS, OR FILE AVAILABILITY
6	SECURITY VIOLATION
7	INADEQUATE ADABAS RESOURCES
8	DATABASE OR FILE SPACE PROBLEM
9	POSSIBLE DATABASE INTEGRITY PROBLEM
A	UNDOCUMENTED RESPONSE CODE

```

EJECT
COPY VERHEAD
EJECT
RSPCLASS CSECT
DS 0CL256      RSP
DC X'00'      0
DC X'07'      1
DC X'07'      2
DC X'00'      3
DC X'01'      4
DC X'07'      5
DC X'07'      6
DC X'04'      7
DC X'07'      8
DC X'04'      9
DC X'01'     10
DC X'0A'     11
DC X'0A'     12
.
.
DC X'05'     240      @V35011
DC X'05'     241      @V35011
DC X'05'     242      @V35011
DC X'05'     243      @V35011
DC X'05'     244      @V35011
DC X'05'     245      @V41048
DC X'0A'     246
DC X'0A'     247
DC X'0A'     248
DC X'0A'     249
DC X'0A'     250
DC X'09'     251
DC X'09'     252
DC X'09'     253
DC X'09'     254
DC X'07'     255
DS 0D
END

```

Data Collector Virtual Storage Requirements

This appendix discusses the factors which determine how much virtual storage Unicenter CA-APAS Insight Monitor for Adabas (Unicenter CA-APAS) Data Collector requires in a given execution.

Factors

Data Collector virtual storage requirements at any point during execution depend on the factors shown below:

ITEM	BATCH MODE	MPM MODE
Code	132K	132K
Globals and main stack	32K	32K
Subtask stack	N/A	2 * 16K
Subtask communication buffer (ENSU-BUFFER)	N/A	as specified
Request, file control blocks	2K each	2K each
Output buffer (per file)	2 * blksize	2 * blksize
Temporary during FLIPXY	32K	
INSIGHT-LINES (per request)	N/A	(no. lines) * line-length
Basic Summary Table		
(one per active SUMMARIZE)		
Index	$12*y/0.86$	$12*y/0.86$
Data	$n*(k+f)$	$n*(k+f)$
Grand-total Table		
(one per active SUMMARIZE w/INTERVAL and GTT=YES)	same as Basic	same as Basic
User ID Table (one per run)	same as Basic	same as Basic

The variable symbols used above are defined as:

- n = number of entries in table (one per unique combination of BY values)
- y = next power of 2 greater than n
- k = key length (combined lengths of BY fields)
- f = field lengths of summary fields and functions

A simple guideline for planning purposes is that 300K bytes should be considered the minimum practical amount of virtual storage for a modest number of requests, while 500K bytes is usually adequate for what most sites typically wish to do.

At the end of each execution, the Data Collector prints a message showing the highest total amount of virtual storage it used during the execution. In general, the amount of space required for execution depends mostly on:

- The number of output files and the blocksizes of these files
- The number of active SUMMARIZE requests and the number and lengths of entries built in the tables constructed for each request

Each of these factors is discussed further below.

Output Buffers

If no blocksizes are specified, output blocksizes default to 4096. Evaluate blocksize against the number of records written to the file. The following general guidelines apply:

- Relatively few records are written for SUMMARIZE requests, so blocksize is usually unimportant
- COPY files could receive a record for every command, so large blocksizes may be very important
- Blocksize is important for an EXTRACT request if it selects all or a large percentage of commands

The price you pay for larger blocksizes is increased use of virtual storage. In virtual storage systems, this is usually a small price to pay, particularly when creating sequential files. Since the data is being added to the buffer sequentially (rather than randomly), the impact on paging is negligible. In fact, large blocksizes improve overall system performance by reducing the amount of time spent executing access method code and by reducing device contention.

SUMMARIZE Request Table Sizes

One virtual storage summary table is maintained for each active SUMMARIZE request. An additional summary table, equal in size to the first, is maintained if the request includes the INTERVAL and GTT=YES parameters. An entry is constructed in each summary table for each lowest-level combination of break (BY) values. For example, a request such as the following with one hundred jobs over the course of the day accessing an average of four files each generates 400 entries.

```
SAMPLSUM: SUMMARIZE  
COUNT SUM(TOTAL-IO) MEAN(TOTAL-IO) MAX(TOTAL-IO)  
MEAN(DURATION) MAX(DURATION) MEAN(ENQ-TIME) MAX(ENQ-TIME)  
BY JOBNAME BY FILE...
```

Very detailed breakdowns, for example, BY NAT-MOD-ID BY TERM BY FILE BY COMMAND-CODE..., may require relatively large amounts of space because of the large number of unique entries.

The size of each entry is the total of the lengths of the accumulators for each summary field or function, plus the combined lengths of the break (BY) fields. Additional space is required for pointers, counters, etc. The example shown above, with eight summary functions, would require in the range of seventy bytes per entry, or a total of over 30K bytes for the table when no other table management components are involved.

Space is allocated for each summary table when the request is activated. If the initial allocation is exceeded, the table size is increased if enough virtual storage is available; otherwise an implicit RESET occurs (data in the table is output, then all accumulators are reset to zero and processing continues).

The space occupied by a table is released and is available for reuse when the request is deleted. RESET and PAUSE do not release tables. A rough estimate of the space required should be made before executing new summary requests.

When executing the Data Collector in MPM mode, the SHOW command may be issued from the Insight component of Unicenter CA-APAS to monitor the current table sizes for active SUMMARIZE requests. This practice is highly recommended.

Index

&

&RECTYBT, 9-3
&RECTYTP, 9-3
&SUBSYS, 9-3
&SUBTYP, 9-3
&SYSID, 9-3

A

Abend codes
 D37, 4-1
 U0253, 2-14, 3-6
 User 253, 2-14, 3-6
Abnormal Termination, 4-16
ACPYIN, 4-12, 4-13
ACPYOUT, 4-10, 4-12, 4-13
Adabas
 Abnormal Termination, 4-16
Adabas Command Log
 batch mode useage, 2-1
 Dual Files, 2-3
 Parameters, 2-10, 3-4
 Processing, 3-2
Adabas Link Routines, 8-1
 Execution, 8-2
 Standard, 8-3
Adabas MPM Sessions, 2-1, 2-7
ADARES/COPY, 2-2, 2-3
ADARUN

LOGCB, 2-3, 2-10, 3-4
LOGFB, 2-10, 3-4
LOGGING, 2-10, 2-15, 3-4
LOGIB, 2-10, 3-4
LOGIO, 2-10, 3-4
LOGRB, 2-10, 3-4
LOGSB, 2-10, 3-4
LOGUX, 2-10, 3-4
LOGVB, 2-10, 3-4
Parameters, 2-10, 2-15, 3-2, 3-4
UEX4, 2-10, 3-4

ADASMFD, 9-2
ADASMFR, 9-2, 9-4
ADASMFY, 9-2, 9-3
Addressing Mode, 5-4, 6-3, 7-3, 8-15
Allocation of Dual Files, 4-3
APASENSU, 2-14
APASJCL DD card, 4-10
APASRDR DD card, 4-10
APASUEX4, 2-10, 3-4, 5-1
ASMSMFY, 9-3

B

Batch Mode
 Advantages, 2-1
 Disadvantages, 2-2
 Input files, 2-2
 Purpose, 2-1
 Restrictions, 2-2
blocksizes, C-2
Buffer Length, 3-3

C

CA-SpaceMan, 2-1
Command Log
 Dual, 2-3
 Parameters, 2-10, 3-4
 Processing, 3-2
Condition Codes, 2-6
COPY files, 2-2
 Sharing, 2-21
COPY Files, 2-17
 DCB information, 2-19
 Format, 2-19
 Mode of Operation, 2-19

D

D37 abend code, 4-1
Data Collector
 batch mode, 1-2
 MPM mode, 1-3, 2-7
 Output, 2-3
 Output Files, 2-11
 Output Record Sequence, 2-3
 OUTPUT-FILE, 2-17
Data Files
 Exit Routines, 2-17
Data Flow, 1-2
Date-time stamp, 2-3
DBGIN, 2-2, 2-5, 2-11, 2-13, 3-5, 4-12
DBGPRINT, 2-5, 2-7, 2-13, 2-15, 2-16, 3-5, 4-12
DCB information, 2-16, 2-17, 2-19, 4-3
DDCARD, 2-13, 3-4
DDCLOG, 2-2, 2-5
DDLOG, 2-13
Derived Fields, 2-18
Derived Fields Exit
 COBOL Exit Routine, 6-4
Derived Fields Exit, 6-1
 Addressing Mode, 6-3
 Input Parameters, 6-2

 Return Parameters, 6-3
Disk Files
 Dual, 2-23
 Single, 2-22
Disk space requirements, 2-22
Driver Modules, 8-6
Driver/Stack routines, 8-3
 benefits, 8-3
Dual Command Log, 2-3
Dual Disk Files, 2-23
DUAL Files
 DCB information, 4-3
Dual Output Files, 4-1
 Allocation, 4-3
 Logic, 4-4
 Specifying, 4-2
Dual processing, 4-1
DUAL-COPY-EXIT, 4-5, 4-10
 Linkage Requirements, 4-7
DUALCPY, 4-5, 4-10, 4-12, 4-13
 ACPYOUT, 4-10
 control statements, 4-12
 executing, 4-11
 Input Files, 4-13
 JCL Example, 4-11
 Output Files, 4-13
 Routine, 4-8
 TIME-STAMP-TEST, 4-9
 TRIM-BANNER, 4-8
 TRIM-PRINT-HEADER, 4-8, 4-10
DUALEXM, 4-5, 4-10
DUAL-FULL-EXIT, 4-7, 4-14
 OS linkage, 4-14
 return codes, 4-14
DUALOPER routine, 4-15

E

ENQ-TIME, 2-2, 2-8, 3-1

ENSU-BUFFER, 2-11

Error Codes, 2-6

Execution

 with Data Collector, 3-2

 without Data Collector, 3-1

Exit Routines, 2-17

F

FLIPXY command, 2-23, 4-2, 4-5

Format Buffer, 3-3

Formatting files, 4-3

G

Generation data set, 2-3

GLOBALS

 ENSU, 2-11

 ENSU-BUFFER, 2-11

 IGNORE-INIT-ERRORS, 2-14, 3-6

 LOG-DEFAULT, 3-3

GTT=YES parameter, C-3

I

I/O errors, 2-6, 2-14

IEBGENER utility, 4-8

IFB-LENGTH, 2-2, 2-8

IGNORE-INIT-ERRORS, 2-14, 3-6

Input Files

 Data Collector, 2-2

 DUALCPY, 4-13

INTERVAL parameter, C-3

IUB-LENGTH, 2-2, 2-8

J

JCLAPASB, 2-14

JES internal reader, 4-11

L

Link Routine Exit

 Adding Routines, 8-15

 Conventions, 8-8

 Driver Modules, 8-6

 Stack Modules, 8-7

 Stack Routines, 8-9

LOGCB parameter, 2-3, 2-10, 3-4

LOG-DEFAULT, 3-3

LOGFB parameter, 2-10, 3-4

LOGGING parameter, 2-10, 2-15, 3-4

LOGIB parameter, 2-10, 3-4

LOGIO parameter, 2-10, 3-4

LOGRB parameter, 2-10, 3-4

LOGSB parameter, 2-10, 3-4

LOGUX parameter, 2-10, 3-4

LOGVB parameter, 2-10, 3-4

M

MAXLINES, 2-22

MAXRECS, 2-22

MPM Mode, 2-7

 Advantages, 2-8

 Disadvantages, 2-9

 Purpose, 2-7

MPM Sessions, 2-7

O

output buffers, C-2

Output Files, 2-11

- Dual, 2-23, 4-1
- DUALCPY, 4-13
- Sharing, 2-20
- Single, 2-22

Output Record Header, 2-21, A-1

OUTPUT-EXIT Routines, 2-17, 7-1

- Addressing Mode, 7-3
- Input Parameters, 7-2
- Return Parameters, 7-2

OUTPUT-FILE, 2-16, 4-2

- Format, 2-17
- Output Record Header, 2-21
- parameter, 2-17
- Record Header, A-1
- Sharing, 2-21
- TIME-STAMP-TEST, 4-13

OUTPUT-FILES

- DCB information, 2-17

P

Performance History System, 1-5

Print Record Header, 2-16, 2-21, 4-3, 4-8, 4-10, A-5

PRINT TO files, 2-16, 4-2

- Carriage control information, 2-16
- DCB information, 2-16
- Print Record Header, 2-21, A-5
- Sharing, 2-20

R

Record Buffer, 3-3

Region size, 2-11

Response Codes, 2-6

Restrictions

- Batch Mode, 2-2

RSPCLASS CSECT source code, B-1

S

Search Buffer, 3-3

Sequence of Output Records, 2-3

Sequential Command Log, 2-3

Sharing

- COPY files, 2-21
- Output files, 2-20
- OUTPUT-FILE files, 2-21
- PRINT TO files, 2-20

Single disk files, 2-22

SIZE parameter, 2-11

SMF

- &RECTYBT, 9-3
- &RECTYTP, 9-3
- &SUBSYS, 9-3
- &SUBTYP, 9-3
- &SYSID, 9-3
- ADASMFD, 9-2
- ADASMFR, 9-2, 9-4
- ADASMFX, 9-2, 9-3
- ASMSMFX, 9-3
- Header fields, 9-2
- SMFRECRD, 9-2, 9-4

SMFRECRD, 9-2, 9-4

Stack Modules, 8-7

Statements

- DUALCPY, 4-12

Storage Requirements, 3-5

sub-task communication buffer, 2-11

T

TIME-STAMP-TEST, 4-12

- Parameter, 4-9

TRIM-BANNER

- Parameter, 4-8

TRIM-PRINT-HEADER, 4-12

- Parameter, 4-8, 4-10

Trouble shooting, 2-7, 2-15

U

U0253 abend code, 2-14, 3-6
UEX4 parameter, 2-10, 3-4, 5-1
UEXITA, 8-2, 8-18
UEXITB, 8-1, 8-2
 Addressing Modes, 8-15
 Conventions, 8-8
 Input Parameters, 8-14
 Return Parameters, 8-15
 Stack Routines, 8-9
User 253 abend code, 2-14, 3-6
User Exit Four, 3-1
 Addressing Mode, 5-4
 Input Parameters, 5-3
 Local Exit, 3-3
 Local Routines, 5-1
 Return Parameters, 5-4

UEX4 parameter, 5-1

User Exit Four, 3-3

User Information Area (UIA)
 Size, 8-15

User Information Data Area, 8-12, 8-13

User-Derived Fields Exit
 Addressing Mode, 6-3

USER-EXIT, 6-1

USER-TYPE, 3-1

V

Virtual Storage, 1-3, 3-5
 Requirements, C-1
SUMMARIZE Request Table Sizes, C-3

