Adabas® Bridge for VSAM Manual

Manual Order Number: AVB511-030IBM

This document applies to Adabas Bridge for VSAM Version 5.1 and to all subsequent releases.

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

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

Documentation@softwareag.com

TABLE OF CONTENTS

PREFACE

The documentation for Adabas Bridge for VSAM (AVB) version 5.1 includes this manual and release notes. This manual is organized as follows:

Unit	Contents
Chapter 1	Introduces the concept of Adabas Bridge for VSAM and describes its facilities.
Chapter 2	Tells you how to install Adabas Bridge for VSAM under OS/390 or z/OS.
Chapter 3	Tells you how to install Adabas Bridge for VSAM under VSE/ESA.
Chapter 4	Tells you how to migrate a VSAM application to Adabas
Chapter 5	Tells you how to define the Adabas file structure.
Chapter 6	Tells you how to create a transparency table using the MCTAB macro and its parameters.
Chapter 7	Tells you how to create the options table and set its parameters.
Chapter 8	Discusses the CICS online operation of AVB, including a detailed discussion of AVB logic: how AVB is activated, how it processes requests, and how it is deactivated.
Chapter 9	Tells you how to use AVB online services to operate AVB. AVB online services is written in Natural and runs under CICS.
Chapter 10	Discusses batch operation of AVB, including a detailed discussion of AVB logic: how AVB is activated, how it processes requests, and how it is deactivated.
Chapter 11	Tells you how to modify batch programs to run as Adabas ET logic users to provide Adabas restart/recovery support to VSAM batch applications.
Chapter 12	Discusses ways to debug AVB applications.
Chapter 13	Explains the messages and codes associated with AVB; that is, Adabas LNKENAB messages, AVB messages, ABEND codes, and ET user response codes. Tells you how to interpret AVB online services messages.
Appendix A	Provides COBOL file definitions to illustrate Adabas field definitions and transparency-table parameters.
Appendix B	Provides sample entries for VSAM ESDS and RRDS files.

1

CONCEPTS AND FACILITIES

Adabas Bridge for VSAM (AVB) makes it possible for you to migrate VSAM applications directly to an Adabas database environment without additional programming. Batch and online (CICS) applications that reference VSAM files can be processed against Adabas files.

Adabas Advantages

Migrating to Adabas provides the following benefits:

- Applications can be extended with the powerful indexing facilities of Adabas. You can query, retrieve, and manipulate data using efficient views and paths.
- Applications can be extended with programming languages such as Natural and SQL.
- Application programs are independent of the data structure, which reduces maintenance costs and increases programmer productivity.
- Automatic restart/recovery ensures the physical integrity of the database in the event of a hardware or software failure.
- Data compression significantly reduces the amount of online storage required and allows you to transmit more information per physical I/O.
- Security is improved by password protection at both the file and field levels and, on the basis of data values, at the record level as well.
- Adabas provides encryption options, including a user-provided key that drives the encryption process.

After migration, your application programs have the same view of data as before, but you can structure the new Adabas files to optimize the benefits outlined above.

The tables that identify files to Adabas are external to the applications and may be changed without relinking the application programs. This feature is especially useful when you want to change file or security information, or move applications from test to production status.

How AVB Works

Existing applications that reference VSAM files can be processed against Adabas files. AVB uses a **transparency table** to map names, numbers, and structures of VSAM files to those of corresponding Adabas files.

Once a VSAM file has been migrated to Adabas and defined in the AVB transparency table, it can be **bridged** to Adabas. When a VSAM file is bridged, AVB converts each request for the VSAM file to an Adabas call; the Adabas file is accessed instead of the VSAM file.

Batch Operation

AVB Screening Routine (OS/390)

When AVB is active, an AVB screening routine takes control whenever the application program issues a request to open or close a VSAM file. This routine scans the transparency table to determine whether AVB is to process the request against an Adabas file. If not, the routine passes the request to the operating system to open the referenced VSAM file.

If the request is to be processed against a corresponding Adabas file, the screening routine passes control to another AVB module to open or close the Adabas file. Once the Adabas file has been opened through AVB, all requests to read or update the VSAM file are passed directly to AVB.

AVB B-Transient Routing (VSE/ESA)

When AVB is active, pointers to the standard IBM B-transients are swapped with pointers to AVB B-transients in the SDL. The processing of an application program that makes an open or close request is routed through the AVB B-transients. The DTF-name is compared with the list of DTF-names in the AVB transparency table:

- If a match is found, processing is routed through the AVB open analyzer, which places the address of the AVB request analyzer in the file's VSAM ACB.
- If a match is not found, the IBM B-transients are invoked to process the open or close request.

VSAM Open Simulation

AVB simulates a VSAM open. It allocates VSAM control blocks and inserts information that the application program needs to process results as if they were returned from a standard VSAM file request.

After an Adabas call is processed, AVB returns the results (data and/or response code) to the application using standard VSAM control blocks and work areas. Thus, the translation of the request from VSAM to Adabas is transparent to the application program.

Adabas files can be opened in exclusive-user or ET-user mode.

AVB Batch Logic Diagram

Figure 1–1 illustrates AVB batch logic. A more detailed discussion of this logic is included in the chapter **Batch Operation** starting on page 197.

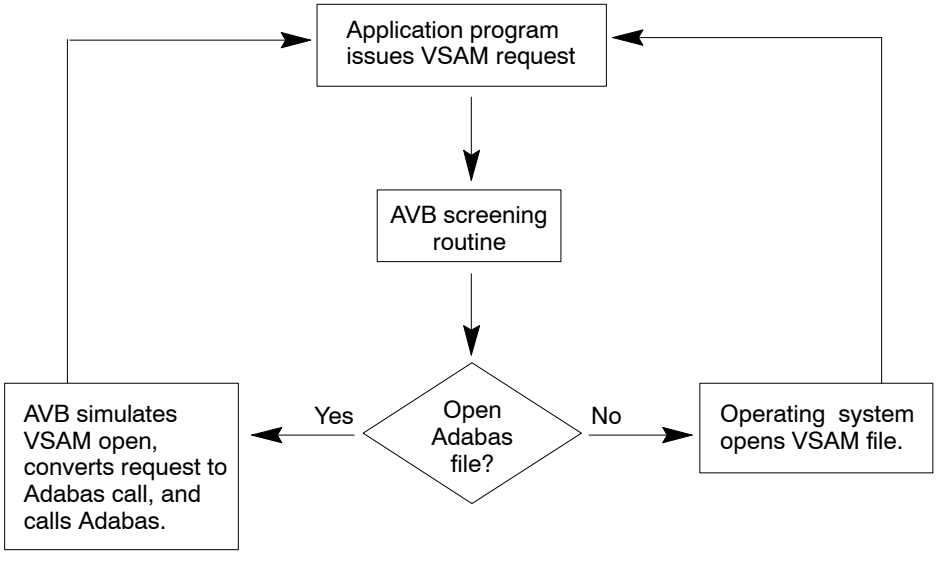


Figure 1-1: Batch Operation of AVB

Online Operation under CICS

AVB is activated and deactivated by

- AVB Online Services;
- a transaction; or
- a program that can be loaded at CICS initialization time.

When AVB is active, it installs a set of CICS global user exits (GLUEs) and a set of CICS task-related user exits (TRUEs). The Adabas command-level link routine with the Adabas TRUE is required for communication between the AVB Adabas request and the Adabas nucleus.

One GLUE program occupies the file control request exits XFCREQ and XFCREQC. The other GLUE program occupies the system programming interface exits XFCAREQ and XFCAREQC. These GLUE points permit AVB to intercept file control requests and system programming interface requests in the CICS address space.

In response to any CICS file I/O request, CICS gives control to the AVB file control GLUE, which scans the AVB transparency table for an entry that matches the DDname (OS/390) or DLBLname (VSE/ESA) provided by CICS on the file I/.O request:

- If a match is found, the request is processed by AVB and the exit sets a return code telling CICS to bypass the request.
- If a match is not found, the request is passed to the CICS file control program for processing by CICS.

If the VSAM “file name” is found in the transparency table and the file is open, the request is processed against the corresponding Adabas file. The request is converted to an Adabas call and the standard Adabas/CICS interface is called to provide the linkage to Adabas. The VSAM return code and any record returned by the Adabas call are returned to the application program through the standard CICS EXEC interface.

System programming interface requests such as INQUIRE FILE and SET FILE are intercepted by the AVB SPI GLUE. Again, if the DDname or DLBLname provided in the request is found in the AVB transparency table, the request is processed by AVB. Otherwise, the request is given to CICS file control for processing.

ET/ BT logic may be activated online by the CICS EXEC SYNCPOINT and CICS EXEC SYNCPOINT ROLLBACK requests. In addition, a syncpoint is taken at CICS END OF TRANSACTION whether the transaction ends normally or abnormally.

If a transaction terminates abnormally, a SYNCPOINT ROLLBACK request is processed by AVB. Syncpoints are handled by a TRUE, which is installed and activated when AVB is activated in CICS:

- An Adabas ET is issued automatically when a transaction terminates normally or requests a CICS SYNCPOINT.
- An Adabas BT is issued automatically if a transaction terminates abnormally or requests a CICS SYNCPOINT ROLLBACK request.

An online transaction is an ET/BT user; therefore, Adabas files are not opened with an OP command.

Figure 1–2 illustrates AVB online logic. A more detailed discussion of this logic is included in the chapter **CICS Operation** starting on page 145.

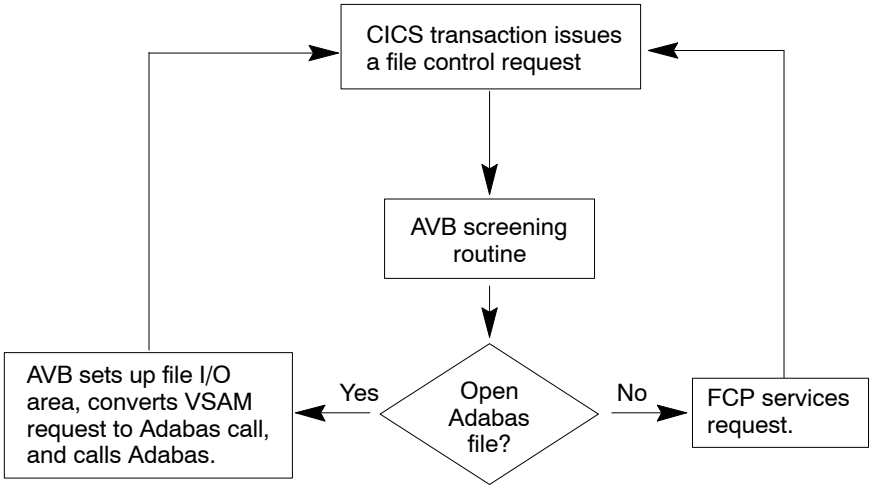


Figure 1-2: Online AVB Operation

Supported File Environments

AVB may be executed in the following file environments:

- Adabas files only;
- VSAM files only;
- Both Adabas and VSAM files (mixed environments).

Note:
Currently, all Adabas files needed for a transaction or an application program must exist in the same physical database.

The ability to operate in a mixed environment means that your migration schedule can be tailored to your needs and resources. You can migrate files from VSAM to Adabas as needed, one application or even one file at a time.

Support for VSAM

VSAM File Types

AVB supports KSDS, ESDS, and RRDS file types.

When using ESDS or RRDS files, follow the same installation and verification procedures as for other VSAM files, but substitute the modules listed on page 20 during the steps for loading test files and verifying the installation.

AVB uses the VSAM relative record number (RRN) as the Adabas ISN. For RRDS files, you must build your own VSAM RRN and will need to use an Adabas user exit 6 in the ADACMP compression step when loading the file. See page 18.

Add CSD entries for the test files you load. Supported file entries are listed on page 35.

A new parameter VSAMTYP must be coded in the transparency table (MCTAB) if you are using ESDS or RRDS files and the PREFSZE parameter must be set to zero (0). See page 104.

Example AVB entries for RRDS (COBOL) and ESDS (PL/I) are provided in appendix B on page 295.

VSAM File Restrictions

The following restrictions apply to VSAM files used with AVB:

- VSAM segmentation is not supported.
- VSAM emulating ISAM is not supported.

VSAM Variable Records

To support VSAM variable records for an AVB file, AVB uses the options table (AVBOPT) parameter VARLRECL to define an additional four-byte binary field at the beginning of every Adabas record in that file. The first two bytes contain the record length; the next two bytes contain low values. This setting must be reflected in the transparency table (MCTAB): the Key Offset field for that AVB file will be offset four bytes to include the length of the Length field itself, and the record size will be similarly affected.

The AVBOPT parameter VARLRECL=ALL supports VSAM variable records for **all** AVB files; while VARLRECL=ON supports VSAM variable records for certain AVB files only. In this case, the transparency table (MCTAB) parameter VARLRECL must also be set to ON for each file that will have support for VSAM variable records.

VSAM Access Options

Macros and Control Blocks

The VSAM access options supported by AVB include the following:

- MODCBs for the ACB, RPL, and EXLST control blocks are supported as long as the modified options are supported.
- The TESTCB and SHOWCB macros can be used for fields or values that have meaning in the AVB environment.
- The ENDREQ control block is accepted but causes no action.
- EODAD and LERAD exits specified in the EXLST control block are supported.
- TCLOSE is not currently supported for VSE/ESA and is not supported in the CLOSE control block for OS/390.

ACB Options

The table below shows which MACRF and RPL functions in the ACB (and their equivalent for high-level languages) are currently supported:

Note:
ACB option RMODE31=ALL is now supported.

Supported MACRF Options

AVB supports the following MACRF options in the access method control block (ACB):

Option	Description
DIR	Direct processing
IN	Retrieve only
KEY	Key access
NDF	Write operations are not to be deferred
NRM	File specified in DDNAME is to be processed
NRS	Open does not reset the file to empty
NSR	Nonshared resources
NUB	No user buffers
OUT	Retrieve, insert, update, delete
RST	Open resets the file to empty
SEQ	Sequential processing
SKP	Skip sequential processing

Unsupported MACRF Options

AVB **does not** support the following MACRF options in the ACB:

Option	Description
ADR	Addressed access
CNV	Control-interval access
DFR	Write operations deferred when possible
LSR	Local shared resources
UBF	User buffers

Supported RPL Options

AVB supports the following options in the RPL control block:

Option	Description
ARD	Search argument provided in ARD determines the record to be processed
BWD	Backward processing supported with “native” Adabas read-logical backward under Adabas version 6.2 and above.
DIR	Direct processing
FKS	Full key
FWD	Forward processing
GEN	Generic key
KEY	Key access
KEQ	Key equal
KGE	Key greater or equal
LOC	Processing in I/O buffer
MVE	Record is moved from buffer to user work area
NSP	Request is not for update; however, a position points to the next record for sequential processing
NUP	Request is not for update
SEQ	Sequential processing
SKP	Skip sequential processing
UPD	Request is for update

Unsupported RPL Options

AVB **does not** support the following RPL options:

Option	Description
ADR	Addressed access
CNV	Control-interval access
LRD	Point to last record (for BWD)

2

OS/390 INSTALLATION

This chapter describes the installation of Adabas Bridge for VSAM (AVB) for use in OS/390 or z/OS environments.

Lines of source code that must be modified during the installation are indicated in the source members themselves.

The AVB transparency table that is installed during these procedures is used for the installation-verification programs (IVPs). After installing AVB successfully, you must create the transparency table(s) for your own files. The transparency table is described in the chapter **Creating the Transparency Table** starting on page 95.

Overview

Three sets of steps are used to describe the installation of AVB. Step sets 1 and 2 are used to install AVB in batch environments; steps 1 and 3 are used to install AVB in CICS environments.

1. Steps common to both batch and CICS environments:

Step	Action
1	Unload the AVB installation datasets to disk.
2	Create the AVB options table (AVBOPT).
3	Create the AVB transparency table (AVBTAB).
4	Install the Adabas user exit 6 routine.
5	Load the AVB test files.

2. Steps for batch environments only:

Step	Action
6	Modify the AVBASM procedure. If you modified AVBASM while installing the Adabas user exit 6 for the COBOL test file, you can skip this step.
7	Install the AVB options table in the batch environment.
8	Install the IVPTABB transparency table for the test files.
9	Copy modules to an APF link-listed library.
10	Verify the AVB installation.

3. Steps for CICS environments only.

Step	Action
6	Install the AVB options table in the CICS environment.
7	Install the AVBTABC transparency table for the test files.
8	Modify the Adabas CICS command-level link routine.
9	Install the AVBCICS4 table.
10	Add AVB entries to the CICS PLTPI and PLTSD tables.
11	Define AVB programs, tables, transactions, and test files to CICS.
12	Install AVB online services.
13	Verify the AVB installation.

Common Steps

Step 1. Unload the AVB Datasets to Disk

The AVB installation tape is a standard-label tape. The *Report of Tape Creation* that accompanies the installation tape lists the volume serial number, density, media type, datasets, and dataset sequence numbers.

The tape is compatible with Software AG's System Maintenance Aid (SMA). For information about using SMA in the installation process, consult the SMA manual.

Contents of the Installation Tape

The tape contains the datasets described below, where ‘s’ indicates the system maintenance (SM) level of AVB version 5.1. Space requirements are based on a 3380 disk device type.

Name	Content	Space
SMT121.TABS	SMA dataset	
AVB51s.LOAD	AVB load library	4 CYL
AVB51s.SRCE	AVB source library	2 CYL
AVB51s.JOBS	AVB jobs library	1 CYL
AVB51s.INPL	AVB Online Services INPL file	5 CYL
AVB51s.ERRN	AVB Online Services messages file	1 CYL

The datasets were loaded to tape using the IBM utilities IEBCOPY and IEBGENER. The *Release Notes* provides a complete listing of the distributed libraries.

The JCL members in the AVB 5.1.s source library include the following variables:

vvvvvv	volume serial number
uuuu	disk unit
tttt	tape unit
vrs	AVB version, revision, and SM level

In many of the members, you will also need to modify dataset names, JOB card accounting information, and other values. Adabas DDCARD and DDKARTE input data must be modified or supplied to meet your installation’s needs.

AVB Jobs Library

To aid in the preparation of JCL at your site, a jobs library has been included in dataset AVB51s.JOBS on the installation tape.

The AVB51s.JOBS members must be modified to suit your site requirements. Values that must be supplied or changed are documented in the members themselves or in the instructions for installation steps where the jobs are used.

The following table describes the jobs in AVB51s.JOBS. IVPs, or **installation-verification programs**, are sample COBOL and PL/I programs used to test AVB operation.

Procedure	Usage
AVBASM	Assembling the AVB batch components
ASMCIVP	Preprocessing, assembling, and linking the CICS IVPs
ASMCIVR	Preprocess, assemble, and link the CICS command-level IVP for RRDS
AVBLOD5	Loading the IVP test files into Adabas
COBAVB	Compiling the COBOL IVPs
XAVBCBL	Executing the COBOL IVPs
PLIAVB	Compiling the PL/I IVPs
XAVBPLI	Executing the PL/I IVPs
RUNAVB	Executing AVB in batch (skeleton procedure)

Sample JCL for Unloading the AVB Installation Datasets

If SMA is unavailable, use the following JCL to download the datasets from tape to disk. It specifies the **minimum** sizes needed to download the datasets.

Make the following changes for your site:

Change . . .	To . . .
vvvvvv	the volume serial number of the tape
uuuu	the disk unit for the unloaded datasets
tttt	the tape unit for the cartridge or tape
vrs	the AVB version, revision, and system maintenance (SM) level
p	the position of the dataset on the tape (see the *Report of Tape Creation*)

```
//AVBINST    JOB
//AVBCOPY    EXEC PGM=IEBCOPY
//SYSPRINT   DD  SYSOUT=*
//IN1        DD  DSN=AVBvrs.LOAD,DISP=OLD,UNIT=tttt,
//               VOL=(,RETAIN,SER=vvvvvv),LABEL=(p,SL)
//OUT1       DD  DSN=SAGLIB.AVBvrs.LOAD,DISP=(NEW,CATLG,DELETE),
//               UNIT=uuuu,SPACE=(CYL,(4,,10))
```

```
//IN2         DD  DSN=AVBvrs.SRCE,DISP=OLD,UNIT=tttt,
//                VOL=(,RETAIN,SER=vvvvvv),LABEL=(p,SL)
//OUT2        DD  DSN=SAGLIB.AVBvrs.SRCE,DISP=(NEW,CATLG,DELETE),
//                UNIT=uuuu,SPACE=(CYL,(2,,15))
//IN3         DD  DSN=AVBvrs.JOBS,DISP=OLD,UNIT=tttt,
//                VOL=(,RETAIN,SER=vvvvvv),LABEL=(p,SL)
//OUT3        DD  DSN=SAGLIB.AVBvrs.JOBS,DISP=(NEW,CATLG,DELETE),
//                UNIT=uuuu,SPACE=(CYL,(1,,5))
//SYSIN       DD  *
COPY IN=IN1,OUT=OUT1
COPY IN=IN2,OUT=OUT2
COPY IN=IN3,OUT=OUT3
//*
//AVBINPL     EXEC PGM=IEBGENER
//SYSIN       DD  DUMMY
//SYSPRINT    DD  SYSOUT=X
//SYSUDUMP    DD  SYSOUT=X
//SYSUT1      DD  DISP=SHR,DSN=AVBvrs.INPL
//SYSUT2      DD  DISP=SHR,DSN=SAGLIB.AVBvrs.INPL
//*
//AVBERRN     EXEC PGM=IEBGENER
//SYSIN       DD  DUMMY
//SYSPRINT    DD  SYSOUT=X
//SYSUDUMP    DD  SYSOUT=X
//SYSUT1      DD  DISP=SHR,DSN=AVBvrs.ERRN
//SYSUT2      DD  DISP=SHR,DSN=SAGLIB.AVBvrs.ERRN
//*
```

Step 2. Create the AVB Options Table (AVBOPT)

The AVB **options table** defines the AVB system dependencies and user options for your site. It is required in all environments where AVB is used.

For information about converting an options table from an earlier version of AVB, or about creating a new options table, see the chapter **Creating an Options Table** starting on page 133.

Step 3. Create the AVB Transparency Table (AVBTAB)

The AVB **transparency table** maps VSAM records referenced in application programs to corresponding Adabas files.

For information about converting a transparency table from an earlier version of AVB, or about creating a new transparency table, see the chapter **Creating a Transparency Table** starting on page 95.

Step 4. Install the Adabas User Exit 6 Routine

AVB uses the VSAM relative record number (RRN) as the Adabas ISN. If you build your own VSAM RRN (and thus a user ISN for the Adabas file), you may need to use an Adabas user exit 6 in the ADACMP compression step when loading the file.

For the COBOL IVP, this procedure is required for KSDS because it is a multirecord type file. For RRDS, it is required for both the COBOL and PL/I IVPs. It uses the following members of the AVB version 5.1.s source library (RRDS-specific module names end with the character 'R'):

Member	Description
COBUEX6	COBOL user exit routine that loads KSDS, the multirecord type file.
COBUEX6R	COBOL user exit routine that builds the Adabas user ISN.
COBUSERT	Assembler-language interface required for the user exit.
COB2ENV COB2ENVR	Assembler-language front-end for the user exit; builds the COBOL/LE environment.
ASMUEX6 ASMUEX6R	JCL to assemble/compile and link the user exit routine. These jobs use AVBASM and COBAVB from the AVB 5.1.s jobs library.

To install the user exit 6 routine

1. Customize member ASMUEX6 and/or ASMUEX6R by supplying job information and modifying other JCL statements as necessary.
2. Customize the COBAVB procedure by supplying values for the symbolic parameters and modifying other JCL statements as necessary.
3. Customize the AVBASM procedure by supplying values for the symbolic parameters and modifying other JCL as required. Note the following:
   - Do **not** supply values for the symbolic parameters **M** and **L**. These values are supplied in the jobs that use the procedure.
   - Specify the AVB source library first in the Assembler SYSLIB concatenation.
   - Do **not** use Assembler F for AVB version 5.1 modules. Use Assembler H or the high-level Assembler.
4. Run ASMUEX6 and/or ASMUEX6R to assemble and link the user exit(s).

Step 5. Load the AVB Test Files

The AVB version 5.1.s jobs library contains the following jobs to load test files:

Job	Language
AVBC5LOD	COBOL KSDS
AVBC5LDR	COBOL RRDS
AVBP5LOD	PL/I KSDS
AVBP5LDE	PL/I ESDS
AVBP5LDR	PL/I RRDS

To load the Adabas test file for each language used at your site

1. Modify the AVBLOD5 member in the AVB jobs library. Supply values and review other JCL as documented in the source member itself.

   Notes:

   1. *COBOL files only: Ensure that the load library that contains the ADAUEX6 load module is included in the STEPLIB concatenation for the ADACMP COMPRESS step and that the STEPLIB is **not** APF-authorized. Otherwise, an ABEND can occur.*
   2. *Under OS/390 version 2.4 and above, it is necessary to use the LE.SCEERUN runtime library in the STEPLIB concatenation for the ADACMP COMPRESS step to avoid IGZ0011C initialization errors when the ADAUEX6 program is invoked while loading the COBOL IVP files.*

2. Modify the DDCARD member in the AVB source library and supply the required values.
3. Modify the source member for **each** language used at your site. Supply values and review other JCL as documented in the source member itself.
4. Run each modified job.

ESDS/RRDS Support

Substitute the modules listed below during the steps for loading test files:

Member	Function	Language
AVBCIVPR	CICS command-level IVP for RRDS	Assembler
ASMCIVR	Preprocess, assemble, and link the CICS command-level IVP for RRDS	
AVBC5LDR	Load the RRDS test file	COBOL
AVBCCOMR	Compile the IVP for RRDS	
AVBCEXER	Execute the IVP for RRDS	
AVBP5LDE	Load the ESDS test file	PL/I
AVBP5LDR	Load the RRDS test file	
AVBPCOME	Compile and link the IVP for ESDS	
AVBPCOMR	Compile and link the IVP for RRDS	
AVBPEXEE	Execute the IVP for ESDS	
AVBPEXER	Execute the IVP for RRDS	

Steps for Batch Only

Step 6. Modify the AVBASM Procedure

This procedure is used in assembling and linking AVB tables and components for a batch environment. It is provided in the AVB jobs library.

To modify the AVBASM procedure

Modify the AVBASM source member. Supply values for the symbolic parameters and modify other JCL as required.

- Do **not** supply values for the symbolic parameters M and L. These values are supplied in the jobs that use the procedure.
- Specify the AVB source library first in the assembler SYSLIB concatenation.

Important:
Use Assembler H or the high-level Assembler. Do not use Assembler F for AVB 5.1 modules.

Step 7. Install the Options Table

To install the options table in a batch environment

1. Modify the source member ASMOPT. Supply job information.
   - The load module name **must** be AVBOPT.
   - Do not use the linkage-editor parameters RENT or REUS.
   - The AVBOPT load module's RMODE can be either 24 or ANY.
2. Run ASMOPT.

   Note:
   Use ASMOPT to reassemble and relink the options table each time you change it.

Step 8. Install the Transparency Table for the Test Files

The IVP transparency table maps the VSAM and Adabas test files bridged by the IVPs. After verifying the installation, you can create entries for your production files and then reassemble and relink the transparency table.

To install the IVP transparency table in the batch environment

1. Modify the member IVPTABB in the AVB version 5.1 source library.

   In the MCTAB TYPE=INITIAL statement, supply a value for the DBID (database number). This value must match the value specified in the ADARUN DBID statement in the DDCARD source member, which was used to load the test files.

   In the MCTAB TYPE=GEN entry for each test file you loaded, supply a value for the FNR (file number); this value must match the FILE value specified in the ADALOD step of the job used to load the file.
2. Modify ASMTABB in the jobs dataset.

   Supply job information and modify other JCL statements as required for your installation.
3. Run the job ASMTABB.

   Note:
   Use ASMTABB to reassemble and relink the transparency table each time you change it.

Step 9. Copy Modules to an APF Link-Listed Library

You must copy a number of modules to an APF link-listed library so that the AVBLOAD program can establish the necessary AVB environment:

- the supplied modules AVBESTAE, AVBETBT, AVBCOMIT, AVBHOOKC, AVBHOOKO, AVBHOOKR, AVBLOAD, AVBRESMG, AVBRIDGE, AVBSCRN, and AVBSOCNM.

  Note:
  Ensure while copying these modules that their aliases are preserved.

- the user-created modules AVBOPT and AVBTAB.

To copy the modules

1. Modify the source member AVBCOPY.
2. Run the job AVBCOPY.
3. Activate the modules.

   Ensure that the new modules are available to the operating system. For example, it may be necessary to issue the command

   F LLA,REFRESH

Step 10. Verify the Batch Installation

To compile, link, and execute the batch IVPs

1. Modify the procedures (procs) used to compile and execute the IVPs for the language(s) used at your site:

Job	Language	Usage
COBAVB	COBOL	Compile
PLIAVB	PL/I	Compile

The procs are included in the AVB version 5.1 jobs library.

Supply values for the symbolic parameters in the procs and modify other JCL statements as required.

Note:
*Do **not** code a value for the TRANS symbolic parameter.*

2. Supply job information in the AVB source member for each language used at your site:

Member	Language
AVBCCOM	COBOL
AVBPCOM	PL/I

3. Run the jobs to compile the IVPs.

4. Check the AVB source member DDCARD to ensure that the DBID, Adabas SVC number, and disk device type are correct.

5. Supply job information in the AVB source member for each language used at your site:

Member	Language
AVBCEXEC	COBOL
AVBPEXEC	PL/I

Note:
*Each member has a symbolic override for the TRANS parameter; it identifies the AVB source member that contains the transaction data for a particular IVP. Do **not** modify this value. An incorrect value will lead to improper execution of the IVP. Each source member states the correct TRANS value.*

You can modify the AVBUSER parameters, although no modification is required. See the discussion of AVBUSER statements in chapter **Batch Operation** starting on page 197.

6. Run the jobs in batch and review the output.

Steps for CICS Environments Only

Step 6. Install the AVB Options Table

To install the options table in a CICS environment

1. Modify the source member ASMOPT. Supply job information. Note the following:
   - The load module name **must** be AVBOPT.
   - Do **not** use the linkage-editor parameters RENT or REUS.
   - The AVBOPT load module's RMODE can be either 24 or ANY.
2. Run ASMOPTC.

   Note:
   Each time you change the options table, you must reassemble and relink it for the CICS environment using ASMOPT.

Step 7. Install the Transparency Table for the Test Files

The IVP transparency table maps the VSAM and Adabas test files bridged by the IVPs. After verifying the installation, you can create entries for your production files and then reassemble and relink the transparency table.

To install the IVP transparency table

1. Modify the AVB version 5.1 source member AVBTABC.

   In the MCTAB TYPE=INITIAL statement, supply a value for the DBID (database number). This value must match the value specified in the ADARUN DBID statement in the DDCARD source member that was used to load the test files.

   In the MCTAB TYPE=GEN entry for each test file you loaded, supply a value for the FNR (file number); this value must match the FILE value specified in the ADALOD step of the job used to load the file.

2. Modify the source member ASMTABC.

   Supply job information and modify other JCL statements as required for your installation.

3. Run ASMTABC.

   Note:
   Each time you change the transparency table, you must use ASMTABC to reassemble and relink it for the CICS environment.

Step 8. Modify the Adabas CICS Link Routine

Command-Level Required

AVB version 5.1 requires an Adabas version 7.1 or above CICS command-level link routine and task-related user exit (TRUE). Use the routine and TRUE from the Adabas CICS library delivered with Adabas. See the Adabas documentation for information about installing the link routine and TRUE.

A macro-level link routine or an earlier version of the command-level link routine **cannot** be used with AVB version 5.1.

To prepare the Adabas CICS command-level link routine for use with AVB

1. Name the command-level link routine and the task-related user exit modules used with AVB 5.1.

   Determine the names you will use for the command-level link routine and the task-related user exit that will be associated with it. You must use these names consistently in the ADAGSET parameters in the LNKOLSC, LNKTRUE, and LNKENAB source members.

   In each member, the ADAGSET parameter ENTPT must specify the name of the command-level link; this name must match the value specified for the parameter ADALNAM of the AVB options table.

   In each member, the ADAGSET parameter TRUENM must specify the name of the task-related user exit; this name must match the value specified for the parameter ADATNAM in the AVB installation options table (MCOPT).

   There is a one-to-one relationship between the command-level link routine and the task-related user exit. Thus, if you are installing multiple instances of the command-level link routine, each pair of names **must** be unique.

2. Modify the ADAGSET parameter overrides in the macro library used to assemble the Adabas CICS command-level components.

   Software AG recommends that you modify a copy of ADAGSET and use it with each instance of the Adabas CICS command-level components; that is, LNKENAB, LNKTRUE, LNKOLSC, and LNKOLM. The ADAGSET macro in these components is now supplied without prototype values to encourage you to modify the ADAGSET macro centrally.

   You do **not** need to modify the equates in LNKOLSC.

ADAGSET Overrides

Required

The following ADAGSET overrides must be set for AVB:

AVB={**YES** | **NO**}

Specify whether the link routine is used for AVB processing.

Failure to code AVB=YES in a link routine used by AVB may cause ABENDs of CICS transactions and functions during OPEN or CLOSE processing.

ENTPT={entry-point-name | **ADABAS**}

Specify the entry-point name for the link routine supporting AVB.

This value must match the value specified in the AVBOPT parameter ADALNAM.

LOGID=dbid

Specify the number of the default database for the link routine.

Two-byte DBIDs are supported; there is no default value.

NUBS={number | **50**}

Specify the number of user blocks to be created by the link routine.

The number of blocks must be sufficient to handle all concurrent CICS users.

PARMTYP={**ALL** | **COM** | **TWA**}

Specify the area in which the link routine picks up the Adabas parameter list to be passed:

ALL in both the COMMAREA and the TWA; the link routine checks the COMMAREA first.

COM in the CICS COMMAREA, which must be at least 32 bytes long. The list must begin with the 8-byte label "ADABAS52".

TWA in the first six fullwords of the TWA.

AVB version 5.1 does not use the TWA, so base the value on other applications that will use this copy of the link routine. If the TWA is used, transactions that invoke the link routine must have a TWASIZE of at least 24 bytes.

PURGE={**YES** | **NO**}

Specify whether CICS can purge a transaction for which the link routine issues a CICS WAIT EXTERNAL.

If PURGE=YES (the default), CICS **can** purge the transaction if the CICS resource definition specifies SPURGE=YES and the CICS DTIMOUT value is exceeded.

For information about CICS WAIT EXTERNAL, consult the *CICS Transaction Server Application Programming Reference* manual.

SVCNO={number | **0**}

Specify the Adabas SVC number. Enter the value appropriate for your system.

TRUE={**YES** | **NO**}

Specify whether the link routine uses the task-related user exit.

TRUE=YES is **required** for AVB version 5.1.

TRUENM={user-exit-name | **ADATRUE**}

Specify the linked name of the task-related user exit, which must match the value specified for the parameter ADATNAM in the AVB installation options table MCOPT. It is critical that the value be provided correctly in both the ADAGSET macro and the MCOPT macro since AVB uses this name to determine whether the Adabas TRUE is active during AVB CICS initialization. If AVB cannot determine that the Adabas TRUE is active, AVB will not initialize under CICS.

UBPLOC={**ABOVE** | **BELOW**}

The UBPLOC parameter determines whether the Adabas user block (UB) pool is allocated above or below the 16-megabyte line in CICS.

Optional

You may choose to set one or more of the following ADAGSET overrides at this time:

ESI={**YES** | **NO**}

Specify whether the task-related user exit passes the user's external security ID (sign-on) to Adabas.

LRINFO={length | **0**}

Specify the length of the data area to be used by the REVEXITB program.

LRINFO=0 (default) indicates that Adabas Review is not used.

For information about setting this value, consult your REVIEW installation documentation.

LUINFO={length | **0**}

Specify the length of the user data to be passed from the CICS link component to Adabas UEXITA and UEXITB.

LUINFO=0 (the default) indicates that no user data is passed.

LUSAVE={length | **0**}

Specify the length of the user save area to be used by Adabas UEXITA and UEXITB. If you specify a value for this parameter, it must be 72 or higher.

LUSAVE=0 (the default) indicates that no user data is passed.

NETOPT={**YES** | **NO**}

Specify whether the 8-byte user ID is constructed from the VTAM LU name.

By default (NETOPT=NO), the user ID is CICSnnnn, where “nnnn” is the CICS terminal ID (for terminal tasks) or task number (for other tasks).

SAP={**YES** | **NO**}

Specify whether the SAP application system is used.

If SAP=YES, the LNKOLSC program creates the user ID for SAP applications from the constant provided in the SAP initialization call plus the field ACBADD2. For more information, consult the supplementary information provided to customers who use the SAP application system.

XWAIT={YES | NO}

Specify whether to generate a standard CICS WAIT (XWAIT=NO) or a CICS WAIT EXTERNAL (XWAIT=YES) in the command-level link routine.

Note that XWAIT=NO was the default before Adabas version 6.1.3. See the *Adabas Installation Manual* for a discussion of the XWAIT parameter and CICS task performance.

Step 9. Install the AVBCICS4 Table

CICS processing requires not only the AVB options and transparency tables, but also the AVBCICS4 table to hold information about AVB CICS transactions that require syncpoint processing.

The table must be resident; it can reside above or below the 16-MB line.

To install the AVBCICS4 table

1. Modify the AVBCICS4 source member, which invokes the MCACTT (for VSE/ESA, MCACTT.A) macro from the AVB source library.

   Supply a value for the ENTRIES keyword on the macro prototype that is large enough to handle all AVB transactions that might be waiting for syncpoint processing in the running CICS region. Because entries in the AVBCICS4 table are cleared when a syncpoint is successfully processed, it is not necessary to consider every AVB transaction that will update VSAM files.

   The ENTRIES value must be greater than or equal to the value of the ADAGSET NUBS parameter used to assemble the Adabas command-level link routine in the same CICS system.

2. Supply job information in source member ASMCICS4.

3. Run ASMCICS4 to assemble and link the table.

Step 10. Add AVB Entries to the CICS PLTPI and PLTSD

Software AG strongly recommends that you use the CICS start-up table (PLTPI) and shut-down table (PLTSD) to activate and deactivate AVB automatically. You **must** use the PLTPI to activate the Adabas task-related user exit (ADAENAB) when CICS is started. If ADAENAB is not executed prior to activating AVB with AVBCICS0, AVB will not initialize in CICS.

Although you can use the AVB1 transaction to activate and deactivate AVB manually, it is intended for use in test environments or to resolve problems in a running CICS region.

Note:
To avoid improper execution of the AVB routines, give the PLTPI and PLTSD tables different suffixes so that they are separate tables.

To add entries if you are already using the PLTPI and PLTSD tables

1. Add entries for the programs AVBCICS0 and ADAENAB in the PLTPI table.

   Put the AVBCICS0 entry in the second section of the table so that it is invoked during the third stage of PLTPI processing. Place it after the entry for ADAENAB to install the Adabas command-level link routine and Adabas TRUE.

   Executing AVBCICS0 before the third stage produces unpredictable results and may prevent CICS from initializing. Executing AVBCICS0 before the Adabas link routine is installed and active causes AVB 5.1 to abort its activation.

2. Add an entry for the program AVBCICS9 in the PLTSD table. AVBCICS9 should be invoked during the first stage of PLTSD processing.

3. Use the CICS procedure DFHAUPLE (modified for your site) to assemble and link the PLTPI and PLTSD tables into a load library concatenated with DFHRPL.

To add entries if you are not currently using PLTPI and PLTSD tables

1. Modify the member AVBPLTPI in the AVB source library. Specify a SUFFIX (xx) value; this value will be referenced in the CICS start-up JCL.

2. Modify the member AVBPLTSD in the AVB source library. Specify a SUFFIX (xx) value; this value will be referenced in the CICS start-up JCL.

3. Use the CICS procedure DFHAUPLE (modified for your site) to assemble and link the PLTPI and PLTSD tables into a library concatenated with DFHRPL.

Step 11. Define AVB Components to CICS

AVB uses the IBM utility program DFHCSDUP to define the AVB components required in CICS. DFHCSDUP can be run against the CSD file in batch or under TSO; CICS can be running or inactive.

The member DEFAVBC in the AVB source library is used as input to the DFHCSDUP utility to define all required AVB programs, tables, transactions, and files to CICS. One of the definitions contained in member DEFAVBC is shown below:

```
DEFINE PROGRAM(AVBCICS2) GROUP(AVB51)
DESCRIPTION(AVB 5.1.s FCP GLOBAL EXIT (XFCREQ/C XFCSREQ/C))
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
```

To define AVB components to CICS using the DFHCSDUP utility with DEFAVBC

1. Use a local editor to carefully modify the DEFAVBC source member.

   - Do **not** modify the existing RELOAD, RESIDENT, CEDF, DATALOCATION, or DATAKEY/EXECKEY attributes.
   - It is the responsibility of the customer to ensure that modifications to the DEFAVBC member do not adversely affect properties of AVB components defined in the CSD.

   Modify the entry for the Adabas command-level link routine to supply your site's name for the routine. The name must match the ENTPT value specified in the ADAGSET member.

   If this link routine will be used for other applications in addition to AVB, you may need to specify a TWASIZE of at least 24. The TWASIZE is not a requirement for AVB programs.

   There are optional program definitions for the PLTPI and PLTSD tables. If you have already defined the PLTPI and PLTSD elsewhere in your CSD, **delete** the optional definitions. If the tables are not defined elsewhere in the CSD, modify the sample entries and supply the DFHPLTxx suffixes that you used when you assembled and linked the tables.

   There is also an optional transaction definition for AVCI, which is an alias for the CICS-supplied CECI transaction. The AVCI transaction is required only to test AVB functionality using CECI.

   Modify items such as profile attributes as necessary.

2. Set up a JCL job stream to run DFHCSDUP, specifying DEFAVBC as the SYSIN member.

3. Run the DFHCSDUP utility and review all output messages it produces. Any warnings about the AVB profile, the AVCI transaction, and the DFHPLTxx definitions can be safely ignored.

4. Initialize CICS with a list containing the AVB groups just defined.

5. Use the CEDA transaction to review the AVB resource groups and definitions.

6. Use the CEMT transaction to check the program status of the Adabas CICS command-level link routine.

Required Resource Definitions

CICS CSD Groups for AVB Entries

Software AG recommends that you use two resource groups for AVB version 5.1 components:

- **AVB51** for the program, table, and transaction definitions;
- **AVBFILES** for the definitions of files that will be accessed by AVB.

Two CSD groups permit the use of the CEDA INSTALL command to refresh program and transaction definitions in a running CICS region. If only one group is used, CEDA INSTALL will contend with the AVB file definitions and cause the installation to fail. The group names above are recommended but not required.

PLT Entries

If you do not already have them, add PLT definitions for the PLTPI and PLTSD tables.

Program Entries

AVB requires entries for the following programs and tables in the CSD (formerly the PPT). These entries belong to the group **AVBV5**:

AVBCICS	AVBCICS5	AVBCIVP
AVBCICS0	AVBCICS6	AVBFIXTB
AVBCICS1	AVBCICS7	AVBOPT
AVBCICS2	AVBCICS8	AVBPROG
AVBCICS3	AVBCICS9	AVBTAB
AVBCICS4	AVBCICSV	AVBREQT

In addition, an entry in the same group is required for the command-level link routine. For this, use the name specified in the ENTPT parameter in ADAGSET; this is normally **ADABAS**.

An entry in the same group is also required for the task-related user exit. For this, use the name specified in the TRUENM parameter in ADAGSET; this is normally **ADATRUE**.

The following attribute settings apply to **all** AVB programs and tables:

Attribute	Value
Group	AVB51
Language	Assembler
Reload	No
Usage	Normal
Uselpacopy	No
Status	Enabled
Datalocation	Any
Execkey	Module-dependent

Certain programs and tables **must** be defined as resident. The CEDF attribute is recommended for some programs but is not supported for others. The following table provides residence and CEDF specifications:

Module	Must Be Resident?	CEDF
AVBCICS	No	Optional
AVBCICS0	No	Recommended
AVBCICS1	No	Recommended
AVBCICS2	Yes	Optional
AVBCICS3	Yes	Not supported
AVBCICS4	Yes	Not supported
AVBCICS5	No	Optional
AVBCICS6	No	Recommended
AVBCICS7	No	Recommended
AVBCICS8	No	Recommended
AVBCICS9	No	Recommended

Module	Must Be Resident?	CEDF
AVBCICSV	No	Optional
AVBCIVP	No	Recommended
AVBOPT	Yes	Not supported
AVBPROG	Yes	Not supported
AVBTAB	Yes	Not supported
AVBFIXTB	No	Not supported
AVBREQT	Yes	Optional
ADABAS	Yes	Optional
ADATRUE	Yes	Optional

The CICS PLTPI and PLTSD tables also must be defined as resident.

Note:
*When you need to load a new copy of a resident AVB module, use the CICS online resource definition. Do **not** use CEMT NEW; if you do, the program will be marked as nonresident, and processing results will be unpredictable.*

All other attribute settings are at the discretion of the site.

Transaction Entries

AVB version 5.1 requires entries for the following transactions in the CSD (formerly the PCT). These entries belong to the group **AVB51**.

AVB0	AVC1	AVTR
AVB1	AVIP	AVZP
AVB9	AVIR	AVCI (optional; allows use of CECI to test AVB)

File Entries

AVB version 5.1 requires a CSD or FCT entry for each test file that you loaded into your environment. These entries belong to the group **AVBFILES**.

Note:
*Each file to be processed through AVB under CICS **must** be defined as CLOSED,UNENABLED. This definition ensures that the file is not opened before AVB is activated.*

- **COBOL**

SYS021	COBOL KSDS file
SYS0211	COBOL KSDS file; alternate index
SYS0212	COBOL KSDS file; alternate index
SYS0213	COBOL KSDS file; alternate index
SYS021R	COBOL RRDS file

  .
  .

- **PL/1**

PLITEST	PL/1 KSDS file
PLIAIX1	PL/1 KSDS file; alternate index
PLIAIX2	PL/1 KSDS file; alternate index
PLIESDS	PL/1 ESDS file

  .
  .
  .

PLIRRDS	PL/1 RRDS file

  .
  .

Step 12. Install AVB Online Services

You can load the programs and messages for AVB online services using standard Natural INPL and ERRLODUS procedures:

- Use the Natural INPL utility to load the AVB online services programs into the Natural system file. The programs are loaded to the Natural application SYSAVB.
- Use the Natural ERRLODUS utility to load the AVB online services error messages into the Natural error library. Messages specific to the AVB online services application have numbers with the prefix “AVB”.

Refer to the *Natural Administration Manual* for information about using these utilities and their parameters.

To run the INPL and load the error text

Use the following sample JCL and check the reports produced to ensure that no errors have occurred.

Make the following changes for your site:

Change . . .	To . . .
dbid	ID of the physical database where Adabas is installed
dddd	ID of the physical database where Natural is installed
fileno	file number for the Natural FUSER system file
filnum	file number for the Natural FDIC system file
fnr	file number for the Natural FNAT system file
svc	SVC to be used for the current Adabas session
vrs	indicate the version, revision, and SM level, respectively

```
//AVBINPL JOB
//*******************************************
//* THIS IS AN EXAMPLE NATURAL INPL/ERRLODUS JOB
//* TO LOAD THE AVB ONLINE SERVICES MODULES AND
//* ERROR MESSAGES
//*******************************************
//INPL    EXEC PGM=NATBAT,REGION=2000K,        <—— BATCH NATURAL
// PARM=('MT=0,DBID=dddd,STACK=INPL,MADIO=0,IM=D,INTENS=1',            X
//       'FNAT=(,fnr),FUSER=(,fileno),FDIC=(,filnum)')
//STEPLIB DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR     <—— ADABAS LOADLIB
//        DD DSN=NATURAL.LOAD,DISP=SHR         <—— NATURAL LOADLIB
//CMWKF01 DD DSN=AVBvrs.INPL,DISP=SHR          <—— AVB INPL FILE
//CMPRINT DD SYSOUT=X
//DDPRINT DD SYSOUT=X
//DDCARD DD *
ADARUN PROG=USER,DBID=dbid,SVC=svc,MODE=MULTI  <—— DBID AND SVC—NR
/*
//CMSYNIN DD *
B
FIN
/*
```

```
//*****************************************
//ERRN    EXEC PGM=NATBAT,REGION=2000K,        <—— BATCH NATURAL
// PARM=('MT=0,DBID=dddd,MADIO=0,IM=D,INTENS=1',                        X
//       'FNAT=(,fnr),FUSER=(,fileno),FDIC=(,filnum)')
//STEPLIB DD DSN=ADABAS.Vvrs.LOAD,DISP=SHR     <—— ADABAS LOADLIB
//        DD DSN=NATURAL.LOAD,DISP=SHR         <—— NATURAL LOADLIB
//CMWKF01 DD DSN=AVBvrs.ERRN,DISP=SHR          <—— AVB ERRN FILE
//CMPRINT DD SYSOUT=X
//DDPRINT DD SYSOUT=X
//DDCARD DD *
ADARUN PROG=USER,DBID=dbid,SVC=svc,MODE=MULTI  <—— DBID AND SVC—NR
/*
//CMSYNIN DD *
LOGON SYSERR
ERRLODUS
FIN
/*
```

Step 13. Verify the CICS Installation

AVB provides the AVBCIVP program to verify the installation under CICS/ESA. The AVCI transaction can be used for additional testing.

Using the AVBCIVP Program

To verify AVB under CICS using AVBCIVP

1. Define the AVBCIVP program and the AVIP transaction to CICS.
2. Modify member AVBCASM in the AVB jobs library.

   For OS/390 or z/OS: you do **not** need to supply values for the “L” and “M” symbolic parameters; these are provided in the AVBCIVP member.
3. Supply job information in source member ASMCIVP.
4. Run ASMCIVP to preprocess, assemble, and link the IVP.
5. Execute the transaction AVIP. Information about the completed transaction is displayed on the screen. If AVB has been installed correctly under CICS, the bottom line of the screen should read as follows:

   AVIP – TRANSACTION ENDED NORMALLY

Using the AVCI Transaction

The AVCI transaction provides a quick, flexible way to test VSAM requests that are handled by AVB. AVCI must be defined to CICS as an alias for the CICS-supplied CECI transaction.

To test the installation using AVCI

1. If AVB was not activated at CICS startup, execute the AVB1 transaction. Use the AVB1 commands ON and FILES / OPEN to activate AVB and open the IVP files.

   The AVB1 transaction password is the group password defined in the source member AVBTAB.

2. Use AVCI (an alias for CECI) to issue a START BROWSE command.

   The following example is for the COBOL file SYS021:

   AVCI STARTBR FI(SYS021) RI(&K)

   An error message is returned. Press PF5 and define the following variables:

   &K Specify the key length; this value must match the KEYLEN parameter in the transparency table. For the SYS021 test file, KEYLEN=7.

   &I Specify the record length; this value must match the RECSIZ parameter in the transparency table. For the SYS021 test file, RECSIZ=80.

   Press ENTER several times until the command is completed.

3. Enter the following command to perform a READ NEXT:

   READN FI(SYS021) RI(&K) INTO(&I) LENGTH(&L)

   Perform several READ NEXT commands.

4. Enter the following command to end the BROWSE:

   ENDBR FI(SYS021)

5. Start the transaction again and read using an alternate index:

   AVCI STARTBR FI(SYS021n) RI(&K)
   READN FI(SYS021n) RI(&K) INTO(&I) LENGTH(&L)

   —where “n” is 1, 2, or 3.

3

VSE/ESA INSTALLATION

This chapter describes the steps required to install Adabas Bridge for VSAM (AVB) under VSE/ESA.

Overview

Three sets of steps are used to describe the installation of AVB. Step sets 1 and 2 are used to install AVB in batch environments; steps 1 and 3 are used to install AVB in CICS environments.

1. Steps common to both batch and CICS environments:

Step	Action
1	Unload the AVB libraries to disk
2	Obtain source members from the AVB library
3	Establish a procedure for the AVB libraries
4	Create the AVB options table (AVBOPT)
5	Create the AVB transparency table (AVBTAB)
6	Install the Adabas user exit 6 routine
7	Load the test files

2. Steps for batch environments only:

Step	Action
8	Create the AVBPVT object module
9	Link the AVB modules
10	Install the AVB options table
11	Install the AVB transparency table
12	Verify the AVB batch installation

3. Steps for CICS environments only.

Step	Action
8	Install the AVB options table in the CICS environment.
9	Install the AVBTABC transparency table for the test files.
10	Modify the Adabas CICS command-level link routine.
11	Install the AVBCICS4 table.
12	Add AVB entries to the CICS PLTPI and PLTSD tables.
13	Define AVB programs, tables, transactions, and test files to CICS.
14	Install AVB online services.
15	Verify the AVB installation.

Storage Requirements

The present size of AVB in batch is roughly 80K.

The GETVIS requirement by partition can be calculated using the following formula for each VSE job step:

GETVIS =
Size of AVB +
Size of Transparency Table +
Number of Files *
(4K per OPEN Control Block + 1K per CLOSE Control Block + 4K Buffer + Size of Record Area + 2K for VSAM Control Blocks + PREFETCH Size) +
(MAXFILE * 160 Bytes) +
(MAXTRAC * 61 Bytes)

MAXFILE and MAXTRAC are options table parameters.

Add 2K for each file with alternate indices having duplicate keys.

COBOL Versions

The default for the COBOL compile jobs is now COBOL for VSE and COBOL LE.

Support for VS/COBOL for VSE has been dropped, although you may be able to run your existing VS/COBOL for VSE phases with AVB 5.1 with the proper LIBDEF chaining of the COBOL/LE or COBOL for VSE libraries.

Common Steps

Step 1. Unload the AVB Libraries to Disk

The AVB installation tape is a standard-label tape created using IBM's LIBR BACKUP utility. The *Report of Tape Creation* that accompanies the installation tape lists the volume serial number, density, media type, datasets, and dataset sequence numbers.

A complete listing of the distributed VSE library is provided in the *Release Notes*.

The following space is required for Adabas objects, phases, source, and JCS samples as well as phases linked during the installation process:

	Blocks	3350 Tracks	3380 Tracks	3390 Tracks
AVB Library	1800	128	50	40

The tape is compatible with Software AG's System Maintenance Aid (SMA). For information about using SMA in the installation process, see the SMA manual.

If Software AG's System Maintenance Aid (SMA) is not yet available at your site, you can use the following sample JCS to load the AVB 5.1 library from tape and establish an appropriate MSHP environment for AVB.

To manually copy the AVB library from tape to disk

Change the disk EXTENT and MSHP control statements as required.

Supply the appropriate value in the SYSID parameter.

Make the following changes for your site:

Change . . .	To . . .
xxxxx	an appropriate value in the LDEST parameter
cuu	the appropriate address for the tape unit
p	indicate the position of the library file on the tape (see the *Report of Tape Creation*)
vrs	indicate the AVB version, revision, and SM level, respectively
vvvvvv	the appropriate disk volume for the AVB library unloaded from tape
ssss	the starting location for the disk extent
ttt	indicate the number of tracks

```
* $$ JOB JNM=AVBCOPY,CLASS=0,DISP=D,LDEST=(,xxxxx),SYSID=1
* $$ LST CLASS=A,DISP=D
*
// JOB AVBCOPY COPY DATASETS FROM TAPE TO DISK
// ASSGN SYS005,IGN
// ASSGN SYS006,cuu
// MTC REW, SYS006
* TAPE POSITIONED AT TAPE MARK p
// MTC FSF,SYS006,p
// DLBL AVBLIB,'AVB.Vvrs.LIBRARY',99,365
// EXTENT ,vvvvvv,1,0,ssss,ttt
// EXEC LIBR,PARM='MSHP'
 DEFINE LIB= AVBLIB R=Y
 RESTORE SUBLIB=SMALIB.AVBvrs : AVBLIB.AVBvrs –
         TAPE=SYS006 –
         LIST=YES –
         REPLACE=NO
/*
// MTC REW,SYS006
// ASSGN SYS006,UA
/*
```

```
// ASSGN SYS002,DISK,VOL=vvvvvv,SHR
// DLBL AVBLIB,'AVB.Vvrs.LIBRARY'
// EXTENT SYS002,vvvvvv,1,0,ssss,ttt
// EXEC MSHP
 ARCHIVE AVBvrs –
 COMPRISES 9001–AVB–00 –
 RESOLVES 'SOFTWARE AG–AVBvrs'
 ARCHIVE 9001–AVB–00–vrs –
 RESIDENCE PRODUCT=AVBvrs –
 PRODUCTION=AVBLIB.AVBvrs –
 GENERATION=AVBLIB.AVBvrs
/*
/&
* $$ EOJ
```

Step 2. Obtain Source Members from the AVB Library

The Adabas Bridge for VSAM sublibrary members are listed below. A name extension (a period followed by a single letter) indicates the function of each member:

- .A is used for Assembler code, examples, etc.;
- .C is used for COBOL source members;
- .X indicates job control statements or job streams.

Note:
Because AVB5.1 runs under VSE/ESA 2.1 and above, the sample JCS members are included as .X books in the AVB source library. The older .J books have been dropped.

1. Make the following source members available to CMS, ICCF, or the editor at your installation:

Member	Description
AVBOPT.A	Sample options table (batch/CICS)
AVBCICSG.A	Sample group file table (CICS)
AVBCIVP.A	IVP for CICS
AVBTAB.A	Sample Adabas transparency table (CICS)
DEFAVBC.A	Sample AVB definitions for the DFHCSDUP utility
IVPTAB.A	Sample Adabas transparency table (batch)

Member	Description
ASMCICSG.X	Assemble and link sample group file table (CICS)
ASMCIVP.X	Preprocess, assemble, and link the command level IVP (CICS)
ASMOPTB.X	Assemble options table (batch)
ASMOPTC.X	Assemble options table (CICS)
ASMPVT.X	Assemble the AVB partition vector table (batch)
ASMTABB.X	Assemble and link IVPTAB (batch)
ASMTABC.X	Assemble and link AVBTAB (CICS)
AVBACT.X	Initial installation and activation of AVB (batch)
AVBCCOM.X	Required for COBOL verification
AVBCEXEC.X	Execute COBOL batch IVP
AVBC5LOD.X	Load COBOL Adabas file
AVBLINKB.X	Link AVB components (batch)
AVBLINKC.X	Link AVB components (CICS)
AVBPCOM.X	Required for PL/I verification
AVBPEXEC.X	Execute PL/I batch IVP
AVBP5LOD.X	Load PL/I Adabas file
AVBPROC.X	Create AVB PROC
AVBRCOM.X	Required for RPGII verification
AVBREXEC.X	Execute RPGII batch IVP
AVBR5LOD.X	Load RPGII Adabas file

2. Before running the sample jobs, change the following items to match your site requirements:
   - POWER control card
   - job cards
   - DLBL, EXTENT, ASSGN, VOLUME, etc. statements
   - Adabas SVC, DEVICE, and DBID

Step 3. Establish a Procedure for the AVB Libraries

Run AVBPROC to establish a procedure for the AVB libraries. Review the job for any site-dependent changes that may be required.

Step 4. Create the AVB Options Table (AVBOPT)

The AVB **options table** defines the AVB system dependencies and user options for your site. It is required in all environments where AVB is used.

For information about converting an options table from an earlier version of AVB, or about creating a new options table, see the chapter **Creating an Options Table** starting on page 133.

Step 5. Create the AVB Transparency Table (AVBTAB)

The AVB **transparency table** maps VSAM records referenced in application programs to corresponding Adabas files.

For information about converting a transparency table from an earlier version of AVB, or about creating a new transparency table, see the chapter **Creating a Transparency Table** starting on page 95.

Step 6. Install the Adabas User Exit 6 Routine

AVB uses the VSAM relative record number (RRN) as the Adabas ISN. If you build your own VSAM RRN (and thus a user ISN for the Adabas file), you may need to use an Adabas user exit 6 in the ADACMP compression step when loading the file.

For the COBOL IVP, this procedure is required for KSDS because it is a multirecord type file. For RRDS, it is required for both the COBOL and PL/I IVPs. It uses the following members of the AVB version 5.1.s library (RRDS-specific module names end with the character ‘R’):

Member	Description
COBUEX6.C	COBOL user exit routine that loads KSDS, the multirecord type file.
COBUEX6R.C	COBOL user exit routine that builds the Adabas user ISN.
COBUSERT.A	Assembler-language interface required for the user exit.
COB2ENV.A COB2ENVR.A	Assembler-language front-end for the user exit; builds the COBOL/LE environment.
ASMUEX6.X ASMUEX6R.X	JCS to assemble/compile and link the user exit routine. These jobs use AVBASM.X and COBAVB.X from the AVB 5.1.s library.

To install the user exit 6 routine

1. Customize member ASMUEX6 and/or ASMUEX6R by supplying job information and modifying other JCS statements as necessary.
   - AVB version 5.1 requires high-level Assembler.
   - Specify the AVB library first in the LIBDEF concatenation.
2. Run ASMUEX6 and/or ASMUEX6R to assemble and link the user exit(s).

Step 7. Load the AVB Test Files

The AVB version 5.1.s library contains the following jobs to load test files:

Job	Language
AVBC5LOD	COBOL KSDS
AVBC5LDR	COBOL RRDS
AVBP5LOD	PL/I KSDS
AVBP5LDE	PL/I ESDS
AVBP5LDR	PL/I RRDS

To load the Adabas test file for each language used at your site

1. Modify the AVBLOD5 member in the AVB library. Supply values and review other JCS as documented in the source member itself.

   Note:
   COBOL files only: Ensure that the LIBDEF search chain contains the ADAUEX6 PHASE for the ADACMP COMPRESS job step.

2. Modify the DDCARD member in the AVB library and supply the required values.

3. Modify the source member for **each** language used at your site. Supply values and review other JCL as documented in the source member itself.

   Change the file number in the source member to a number available in your database. Specify the correct database ID (DBID), Adabas SVC number, device type, and procedure for Adabas private libraries.

4. Run each modified job.

ESDS/RRDS Support

Substitute the modules listed below during the steps for loading test files:

Member	Function	Language
AVBCIVPR	CICS command-level IVP for RRDS	Assembler
ASMCIVR	Preprocess, assemble, and link the CICS command-level IVP for RRDS	
AVBC5LDR	Load the RRDS test file	COBOL
AVBCCOMR	Compile the IVP for RRDS	
AVBCEXER	Execute the IVP for RRDS	
AVBP5LDE	Load the ESDS test file	PL/I
AVBP5LDR	Load the RRDS test file	
AVBPCOME	Compile and link the IVP for ESDS	
AVBPCOMR	Compile and link the IVP for RRDS	
AVBPEXEE	Execute the IVP for ESDS	
AVBPEXER	Execute the IVP for RRDS	

Steps for Batch Environments Only

Step 8. Create the AVBPVT Object Module

Run the job ASMPVT to create the AVBPVT object module. Set the NRDYNP parameter to match the number of dynamic partitions you have generated in the VSE/ESA system where you are likely to run AVB. The default is 20. The AVBPVT phase must be available in the SVA when AVB is activated, even if it will be run in static partitions.

For more information, see the comments in the JCS and the AVBPVT.A source member.

Step 9. Link the AVB Modules

Note:
If you already have a version of AVB installed and active, you must run AVBSWI to deactivate AVB before performing this step.

The AVBLINKB job links the B-transients, job control exit (AVBJBXT), installation program (AVBINST), activation program (AVBSWI), and AVB module AVBRIDGE.

To link the AVB modules

1. Consider the following points:
   - The transients $$ BAVBC1 and $$ BAVBO1 must be cataloged in a library chained when the SDL is set.
   - The modules AVBPVT and AVBJBXT must be resident in the shared virtual area.
   - The EXTRN “AVBCARDS” and “AVBMSGS” appears when the AVBRIDGE phase is linked.
2. Review source member AVBLINKB for any site-dependent changes that may be required. If you want to use the batch user exits, add an INCLUDE statement for each batch user exit module you want to install.
3. Run AVBLINKB.

Step 10. Install the AVB Options Table

1. Modify member AVBOPT.A for your installation. As a minimum, set the following parameters:

OPSYS	MAXFILE	ADALNM
CSECT=YES	MAXTRAC	ADATNM
		ADAVER

   Because AVB options are systemwide, Software AG recommends that you define all options for your site at this time. See the chapter **Creating the Options Table** starting on page 133 for more information.

2. Modify source member ASMOPTB.X. Supply job card information and modify other JCS statements as required for your installation.

3. Run ASMOPTB.X to assemble and link the options table.

Step 11. Install the AVB Transparency Table

1. Modify source member IVPTAB.A.

   Include the test files loaded in step 7.

   Comment out entries for files you did not load.

   Change the Adabas DBID and file numbers to the numbers you specified in step 7.

   For more information about the transparency table, see the chapter **Creating the Transparency Table** starting on page 95. However, Software AG recommends that you make only the minimal changes described above during the installation and verification process.

2. Modify source member ASMTABB.X.

   Supply POWER and job card information and modify other JCS statements as required for your installation.

3. Run ASMTABB.X to assemble and link the transparency table.

   By default, the name of the IVPTAB module is changed to AVBTAB when the transparency table is assembled and linked. You may rename the transparency table for batch processing; however, it **must** be named AVBTAB for CICS processing. Software AG recommends that you use the default name AVBTAB while installing and verifying AVB.

Note:
*If you assemble using the high-level assembler, you **must** specify assembler parameter COMPAT(SYSLIST). Otherwise, Adabas RSP 60 is returned when you access any files defined in the AVBTAB.*

Step 12. Verify the Batch Installation

Perform the following steps for each language to be verified:

1. Modify the job AVBxCOM.

   Review the job for any site-dependent changes that may be required.

2. Run AVBxCOM to compile and link the installation-verification program (IVP).

   The default for the COBOL sample job is now COBOL for VSE. If you are compiling with VS/COBOL, refer to the FILE–CONTROL section of the COBOL IVP for changes required to the SELECT statement.

3. Run the job AVBACT.X (shown below) to execute the programs AVBINST and AVBSWI.

   Note:
   This job should not be rerun. You may rerun AVBSWI to deactivate AVB, but not AVBINST. Do not execute the SET–SDL twice.

   This job installs the job control exit and activates AVB.

   See the section **Required Changes to ASI IPL Procedures** on page 54 for more information.

```
* $$ JOB JNM=AVBACT,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB AVBACT JOB TO ACTIVATE AVB
*
* THIS JOB:
*  1. DOES THE SET SDL
*  2. ACTIVATES THE JOB CONTROL EXIT
*  3. ACTIVATES AVB
* THIS JOB IS ONLY FOR INITIAL INSTALL, UNTIL THE IBM ASI
* PROCEDURES CAN BE UPDATED AND AN IPL CAN OCCUR.
*
* THIS JOB MUST BE RUN IN THE BACKGROUND PARTITION FOR A SET SDL
* TO OCCUR.
```

```
*
// EXEC PROC=AVBPROC
SET SDL
$$BAVBC1,MOVE
$$BAVBO1,MOVE
AVBAVBO2,SVA
AVBJBXT,SVA
AVBPVT,SVA
/*
// EXEC AVBINST
// EXEC AVBSWI
/*
/&
* $$ EOJ
```

Note:
Messages appear on SYSLOG during the execution of this job. If AVBSWI fails to activate AVB, a return code of 4 is given.

4. Modify source member AVBxEXEC for each language used at your site.

   Change the Adabas DBID and SVC number in the ADARUN cards and the procedure for Adabas private libraries in the JCS.

   If you are executing with VS/COBOL, change the following line of the JCS

   ```
   // DLBL CARDS,'AVB.WORK.FILE1',0,SD
   ```

   —to

   ```
   // DLBL SYS011,'AVB.WORK.FILE1',0,SD
   ```

   Review the job for other site-dependent changes that may be required.

5. Run AVBxEXEC to execute the IVP.

   Note:
   The COBOL IVP may receive a condition code 4095 with VS/COBOL. This should be ignored.

   The batch installation and verification is now complete.

Required Changes to ASI IPL Procedures

To ensure that AVB is present after an IPL

1. Modify the VSE ASIPROC to include the following entries in the SDL for automatic initialization of AVB:

```
$$BAVBO1,MOVE
$$BAVBC1,MOVE
AVBPVT,SVA
AVBJBXT,SVA
AVBAVBO2,SVA
```

Note:
Adabas Review version 3.2 (and above) applications can be run after the AVB version 5.1 B-transients, job exit, and initialization modules are installed.

VSE/ESA sites that use the new $JOBEXIT facility should use the appropriate "$ JOBEX0n" phase name in place of "AVBJBXT" in the table above. See the section ***Using AVB with Multiple Job Exits Under VSE/ESA*** *on page 55.*

2. Include the JCS to execute AVBINST and AVBSWI in the automatic system initialization (ASI) as shown in the example below:

```
// EXEC AVBINST
// EXEC AVBSWI
```

The program AVBINST installs the Adabas VSAM job control exit; the program AVBSWI activates AVB.

Note:
AVBINST and AVBSWI should be executed only once during the ASI. After the ASI, AVBSWI can be rerun to toggle AVB on and off, but AVBINST must not be run twice.

AVBINST and AVBSWI can be run in any partition, but the SDL must be set in the background partition unless it is run on VSE/ESA version 2.5 or above.

Note:
SYSLST should be assigned when the ASI procedure is executed. If it is not assigned, only console messages are written by AVB. Error messages or PDUMP data that would have been written to SYSLST are skipped. If AVBSWI fails to activate AVB, return code 4 is given.

Now the application program can be executed if adequate * AVBUSER statements and ADARUN control cards have been provided in the batch JCS members.

The job AVBACT.X has been provided for initial installation and verification if an IPL cannot bring in the change to the ASI IPL procedures. AVBACT.X performs the SET SDL, initializes the job exit, and activates AVB.

Users of earlier versions of AVB must perform an IPL to include the latest version of the AVB job control exit. The IPL must be performed immediately after linking AVBJBXT.

Do not relink $$BAVBO1, $$BAVBC1, AVBAVB02, AVBPVT, or AVBJBXT if AVB is active.

If for any reason AVBJBXT is renamed $JOBEXIT, you must restore the previous $JOBEXIT or set a dummy $JOBEXIT (like the $JOBEXIT supplied in the VSE/ESA operating system).

Using AVB with Multiple Job Exits Under VSE/ESA

VSE/ESA versions 1.2 and above provide a facility to support multiple job exits that are active in a system simultaneously. AVB 5.1 automatically detects and takes advantage of this facility if it is installed under VSE/ESA.

For more information on the multiple job exit facility, see the *IBM VSE/ESA Guide to System Functions*.

$JOBEXIT Table

If you use the multiple job exit facility, the $JOBEXIT PHASE must be a table listing the job exits to be invoked by the operating system in turn for each JCS statement processed in a partition. The format of the $JOBEXIT table is as follows:

```
$JOBEXIT CSECT
         DC    CL8'JCLUSEX'      Job exit table header
         DC    CL8'$ JOBEX01'    Job exit PHASE name 1
         DC    CL8'IBMEXIT'      Descriptive name 1
         DC    CL8'$ JOBEX02'    Job exit PHASE name 2
         DC    CL8'AVBJBXT'      Required descriptive name for AVB
         DC    CL8'$ JOBEX03'    Job exit PHASE name 3
         DC    CL8'MYEXIT3'      Descriptive name 3
         DC    XL4'FFFFFFFF'     Required table termination string
         END
```

PHASE Names

There can be up to ten job exit phases ($JOBEX00–$JOBEX09) in the table. Phases are invoked in the numeric order of the last digit of the PHASE name. In the example above, $JOBEX02 is the AVB job control exit phase, which is the second job exit invoked for each JCS statement executed.

Descriptive Names

In addition to its PHASE name, each entry in the table has an eight-byte descriptive name. The descriptive name for the AVB job control exit must be “AVBJBXT”.

Important:
Failure to set the descriptive name “AVBJBXT” for the AVB job control exit prevents the proper activation and execution of AVB.

Coding the “AVBJBXT” descriptive name for a $JOBEX0n phase other than the AVB job control exit may have disastrous effects on a running VSE/ESA system.

SET SDL Statements

The following SET SDL statements are based on the example on the previous page. They illustrate the required SET SDL statements for AVB under VSE/ESA using the job exit table:

```
SET SDL
$$BAVBO1,MOVE
$$BAVBC1,MOVE
AVBAVBO2,SVA
AVBPVT,SVA
$JOBEX02,SVA
```

Note that $JOBEX02 is used instead of AVBJBXT to refer to the AVB job control exit in this case.

Implementing the Multiple Job Exit Facility

To implement the multiple job exit facility under VSE/ESA

1. Code a $JOBEXIT table similar to the sample presented above. Assemble and link the table using the “S” attribute so that it can be loaded into the SVA.
2. Link the various job exit object modules, including the AVBJBXT.OBJ module, using the “S” attribute. Ensure that the PHASE names conform to the required names as indicated in the job exit table (page 55).
3. Run a SET SDL to load the job exit table and its component phases into the SVA.
4. Place the phases in libraries accessible during the ASI procedure, and add SET SDL statements to that procedure to install the job exits automatically during the IPL.

Important:
When VSE/ESA is started, a phase with an entry in the $JOBEXIT table that is not found in the SVA prevents the execution of any job exits coded in the table.

Evaluating Job Exit Processing

To ensure that all job exits perform properly together, it is critically important that you evaluate the processing order and functions of each exit in the $JOBEXIT table. An error in a job exit may prevent the subsequent exits from executing. Also, if a job exit places a nonzero value in R15, the JCS statement being processed when the return code is examined by job control is skipped by all subsequent job exit phases.

Enabling and Disabling Job Exits

On a running VSE/ESA system, you can enable or disable job exits with the following console commands:

JOBEXIT ENABLE/DISABLE

—to enable or disable all job exits in the table; or

JOBEX0n ENABLE/DISABLE

—to enable or disable a particular table entry.

These commands can be issued only in the background.

Displaying Job Exit Status

Issuing the JOBEXIT command with no operands displays the status of each job exit on SYSLOG.

Batch User Exit 6

AVB provides the batch Adabas user exit 6, which may be required for multiple-record-type files. A sample COBOL user exit 6 is provided in both object and source members.

Use the ASMUEX6.X or ASMUEX6R.X sample jobs to compile and link user exit 6 for use with the ADACMP utility.

Multiple-Record-Type Files

An Adabas user exit 6 may be required in the ADACMP compression step when loading a multiple-record-type file.

The sample COBOL exit consists of the following source modules:

Module	Description
ADAUEX6	A program (source module COBUEX6.C) that expands records according to record type before passing them to ADACMP.
COBUSERT	A sample COBOL interface.
COBSTRT	A sample COBOL front-end for VSE/ESA. Assembler stub to initialize the COBOL/LE environment.
COB2NV	A sample COBOL/LE initialization routine.

These sample routines can provide the basis for coding exits for other multiple-record-type files.

Steps for CICS Environments Only

Step 8. Install the AVB Options Table

To install the options table in a CICS environment

1. Modify the source member ASMOPTC.X. Supply job information. Note the following:
   - The PHASE name **must** be AVBOPT.
   - The AVBOPT.PHASE's RMODE can be either 24 or ANY.
2. Run ASMOPTC.X.

   Note:
   Each time you change the options table, you must reassemble and relink it for the CICS environment using ASMOPTC.X.

Step 9. Install the Transparency Table for the Test Files

The IVP transparency table maps the VSAM and Adabas test files bridged by the IVPs. After verifying the installation, you can create entries for your production files and then reassemble and relink the transparency table.

To install the IVP transparency table

1. Modify the AVB version 5.1 source member ASMTABC.A.

   In the MCTAB TYPE=INITIAL statement, supply a value for the DBID (database number). This value must match the value specified in the ADARUN DBID statement in the DDCARD source member, which was used to load the test files.

   In the MCTAB TYPE=GEN entry for each test file you loaded, supply a value for the FNR (file number); this value must match the FILE value specified in the ADALOD step of the job used to load the file.
2. Modify the source member ASMTABC.X.

   Supply job information and modify other JCS statements as required for your installation.
3. Run AVBTABC.X.

Note:
Each time you change the transparency table, you must use ASMTABC.X to reassemble and relink it for the CICS environment.

Step 10. Modify the Adabas CICS Link Routine

Command-Level Required

AVB version 5.1 requires an Adabas version 7.1 or above CICS command-level link routine and task-related user exit (TRUE). Use the routine and TRUE from the Adabas CICS library delivered with Adabas. See the Adabas documentation for information about installing the link routine and TRUE.

A macro-level link routine or an earlier version of the command-level link routine **cannot** be used with AVB version 5.1.

To prepare the Adabas CICS command-level link routine for use with AVB

1. Name the command-level link routine and the task-related user exit modules used with AVB 5.1.

   Determine the names you will use for the command-level link routine and the task-related user exit that will be associated with it. You must use these names consistently in the ADAGSET parameters in the LNKOLSC.A, LNKTRUE.A, and LNKENAB.A source members.

   In each member, the ADAGSET parameter ENTPT must specify the name of the command-level link; this name must match the value specified for the parameter ADALNAM of the AVB options table.

   In each member, the ADAGSET parameter TRUENM must specify the name of the task-related user exit; this name must match the value specified for the parameter ADATNAM in the AVB installation options table (MCOPT).

   There is a one-to-one relationship between the command-level link routine and the task-related user exit. Thus, if you are installing multiple instances of the command-level link routine, each pair of names **must** be unique.

2. Modify the ADAGSET parameter overrides in the macro library used to assemble the Adabas CICS command-level components.

   Software AG recommends that you modify a copy of ADAGSET.A and use it with each instance of the Adabas CICS command-level components; that is, LNKENAB.A, LNKTRUE.A, LNKOLSC.A, and LNKOLM.A. The ADAGSET macro in these components is now supplied without prototype values to encourage you to modify the ADAGSET.A macro centrally.

   You do **not** need to modify the equates in LNKOLSC.A.

ADAGSET Overrides

Required

The following ADAGSET overrides must be set for AVB version 5.1:

AVB={**YES** | **NO**}

Specify whether the link routine is used for AVB processing.

Failure to code AVB=YES in a link routine used by AVB may cause ABENDs of CICS transactions and functions during OPEN or CLOSE processing.

ENTPT={entry-point-name | **ADABAS**}

Specify the entry-point name for the link routine supporting AVB.

This value must match the value specified in the AVBOPT parameter ADALNAM.

LOGID=dbid

Specify the number of the default database for the link routine.

Two-byte DBIDs are supported; there is no default value.

NUBS={number | **50**}

Specify the number of user blocks to be created by the link routine.

The number of blocks must be sufficient to handle all concurrent CICS users.

PARMTYP={**ALL** | **COM** | **TWA**}

Specify the area in which the link routine picks up the Adabas parameter list to be passed:

ALL in both the COMMAREA and the TWA; the link routine checks the COMMAREA first. This is the recommended setting.

COM in the CICS COMMAREA, which must be at least 32 bytes long. The list must begin with the 8-byte label "ADABAS52".

TWA in the first six fullwords of the TWA.

AVB version 5.1 does not use the TWA, so base the value on other applications that will use this copy of the link routine. If the TWA is used, transactions that invoke the link routine must have a TWASIZE of at least 24 bytes.

PURGE={**YES** | **NO**}

Specify whether CICS can purge a transaction for which the link routine issues a CICS WAIT EXTERNAL. The recommended setting is PURGE=NO.

If PURGE=YES, CICS **can** purge the transaction if the CICS resource definition specifies SPURGE=YES and the CICS DTIMOUT value is exceeded.

For information about CICS WAIT EXTERNAL, consult the *CICS Transaction Server Application Programming Reference* manual.

SVCNO={number | **0**}

Specify the Adabas SVC number. Enter the value appropriate for your system. SVCNO=45 is a valid value.

TRUE={**YES** | **NO**}

Specify whether the link routine uses the task-related user exit.

TRUE=YES is **required** for AVB version 5.1.

TRUENM={user-exit-name | **ADATRUE**}

Specify the linked name of the task-related user exit, which must match the value specified for the parameter ADATNAM in the AVB installation options table MCOPT.

It is critical that the value be provided correctly in both the ADAGSET macro and the MCOPT macro since AVB uses this name to determine whether the Adabas TRUE is active during AVB CICS initialization. If AVB cannot determine that the Adabas TRUE is active, AVB will not initialize under CICS.

UBPLOC={**ABOVE** | **BELOW**}

The UBPLOC parameter determines whether the Adabas user block (UB) pool is allocated above or below the 16-megabyte line in CICS.

Optional

You may choose to set one or more of the following ADAGSET overrides at this time:

ESI={**YES** | **NO**}

Specify whether the task-related user exit passes the user's external security ID (sign-on) to Adabas.

LRINFO={length | **0**}

Specify the length of the data area to be used by the REVEXITB program.

LRINFO=0 indicates that Adabas Review is not used.

For information about setting this value, consult your Adabas Review installation documentation.

LUINFO={length | **0**}

Specify the length of the user data to be passed from the CICS link component to Adabas UEXITA and UEXITB.

LUINFO=0 indicates that no user data is passed.

LUSAVE={length | **0**}

Specify the length of the user save area to be used by Adabas UEXITA and UEXITB. If you specify a value for this parameter, it must be 72 or higher.

LUSAVE=0 indicates that no user data is passed.

NETOPT={**YES** | **NO**}

Specify whether the 8-byte user ID is constructed from the VTAM LU name.

NETOPT=NO indicates that the user ID is “CICSnnnn”, where “nnnn” is the CICS terminal ID (for terminal tasks) or task number (for other tasks).

SAP={YES | NO}

SAP=NO indicates that the SAP application system is not used. If SAP=YES is set, the LNKOLSC program creates the user ID for SAP applications from the constant provided in the SAP initialization call plus the field ACBADD2. For more information, consult the supplementary information provided to customers who use the SAP application system.

XWAIT={YES | NO}

XWAIT=NO generates a standard CICS WAIT in the command-level link routine and is the recommended setting. XWAIT=YES generates a CICS WAIT EXTERNAL.

Step 11. Install the AVBCICS4 Table

CICS processing requires not only the AVB options and transparency tables, but also the AVBCICS4 table to hold information about AVB CICS transactions that require syncpoint processing.

The table must be resident; it can reside above or below the 16-MB line.

To install the AVBCICS4 table

1. Modify the AVBCICS4.A source member, which invokes the MCACTT.A macro from the AVB source library.

   Supply a value for the ENTRIES keyword on the macro prototype that is large enough to handle all AVB transactions that might be waiting for sync-point processing in the running CICS partition. Because entries in the AVBCICS4 table are cleared when a syncpoint is successfully processed, it is not necessary to consider every AVB transaction that will update VSAM files.

   The ENTRIES value must be greater than or equal to the value of the ADAGSET NUBS parameter used to assemble the Adabas command-level link routine in the same CICS system.

2. Supply job information in source member ASMCICS4.X.

3. Run ASMCICS4.X to assemble and link the table.

Step 12. Add AVB Entries to the CICS PLTPI and PLTSD

Software AG strongly recommends that you use the CICS start-up table (PLTPI) and shut-down table (PLTSD) to activate and deactivate AVB automatically. You **must** use the PLTPI to activate the Adabas task-related user exit (ADAENAB) when CICS is started. If ADAENAB is not executed prior to activating AVB with AVBCICS0, AVB will not initialize in CICS.

Although you can use the AVB1 transaction to activate and deactivate AVB manually, it is intended for use in test environments or to resolve problems in a running CICS region.

Note:
To avoid improper execution of the AVB routines, give the PLTPI and PLTSD tables different suffixes so that they are separate tables.

To add entries if you are already using the PLTPI and PLTSD tables

1. Add entries for the programs AVBCICS0 and ADAENAB in the PLTPI table.

   Put the AVBCICS0 entry in the second section of the table so that it is invoked during the third stage of PLTPI processing. Place it after the entry for ADAENAB to install the Adabas command-level link routine and Adabas TRUE.

   Executing AVBCICS0 before the third stage produces unpredictable results and may prevent CICS from initializing. Executing AVBCICS0 before the Adabas link routine is installed and active causes AVB to abort its activation.

2. Add an entry for the program AVBCICS9 in the PLTSD table. AVBCICS9 should be invoked during the first stage of PLTSD processing.

3. Use the CICS job DFHAUPLE (modified for your site) to assemble and link the PLTPI and PLTSD tables into a library concatenated with DFHRPL.

To add entries if you are not currently using PLTPI and PLTSD tables

1. Modify the member AVBPLTPI.A in the AVB library. Specify a SUFFIX (xx) value; this value will be referenced in the CICS start-up JCL.

2. Modify the member AVBPLTSD.A in the AVB library. Specify a SUFFIX (xx) value; this value will be referenced in the CICS start-up JCL.

3. Use the CICS job DFHAUPLE (modified for your site) to assemble and link the PLTPI and PLTSD tables into a library concatenated with DFHRPL.

Step 13. Define AVB Components to CICS

AVB uses the IBM utility program DFHCSDUP to define the AVB components required in CICS. DFHCSDUP is run against the CSD file in batch; CICS can be running or inactive.

The member DEFAVBC.A in the AVB source library is used as input to the DFHCSDUP utility to define all required AVB programs, tables, transactions, and files to CICS. One of the definitions contained in member DEFAVBC is shown below:

```
DEFINE PROGRAM(AVBCICS2) GROUP(AVB51)
DESCRIPTION(AVB 5.1.s FCP GLOBAL EXIT (XFCREQ/C XFCSREQ/C))
       LANGUAGE(ASSEMBLER) RELOAD(NO) RESIDENT(YES) USAGE(NORMAL)
       USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
       EXECKEY(CICS) EXECUTIONSET(FULLAPI)
```

To define AVB components to CICS using the DFHCSDUP utility with DEFAVBC

1. Use a local editor to carefully modify the DEFAVBC.A source member.

   - Do **not** modify the existing RELOAD, RESIDENT, CEDF, DATALOCATION, or DATAKEY/EXECKEY attributes.
   - It is the responsibility of the customer to ensure that modifications to the DEFAVBC.A member do not adversely affect properties of AVB components defined in the CSD.

   Modify the entry for the Adabas command-level link routine to supply your site's name for the routine. The name must match the ENTPT value specified in the ADAGSET.A member.

   If this link routine will be used for other applications in addition to AVB, you may need to specify a TWASIZE of at least 24. The TWASIZE is not a requirement for AVB programs.

   There are optional program definitions for the PLTPI and PLTSD tables. If you have already defined the PLTPI and PLTSD elsewhere in your CSD, **delete** the optional definitions. If the tables are not defined elsewhere in the CSD, modify the sample entries and supply the DFHPLTxx suffixes that you used when you assembled and linked the tables.

   There is also an optional transaction definition for AVCI, which is an alias for the CICS-supplied CECI transaction. The AVCI transaction is required only to test AVB functionality using CECI.

   Modify items such as profile attributes as necessary.

2. Set up a JCL job stream to run DFHCSDUP using the AVBCSDUP.X member as a guide. Use a text editor to incorporate the updated DEFAVBC.A data so that it is picked up by SYSIPT.

3. Run the DFHCSDUP utility and review all output messages it produces. Any warnings about the AVB profile, the AVCI transaction, and the DFHPLTxx definitions can be safely ignored.

4. Initialize CICS with a list containing the AVB groups just defined.

5. Use the CEDA transaction to review the AVB resource groups and definitions.

6. Use the CEMT transaction to check the program status of the Adabas CICS command-level link routine.

Required Resource Definitions

CICS CSD Groups for AVB Entries

Software AG recommends that you use two resource groups for AVB version 5.1 components:

- AVB51 for the program, table, and transaction definitions;
- AVBFILES for the definitions of files that will be accessed by AVB.

Two CSD groups permit the use of the CEDA INSTALL command to refresh program and transaction definitions in a running CICS region. If only one group is used, CEDA INSTALL will contend with the AVB file definitions and cause the installation to fail. The group names above are recommended but not required.

PLT Entries

If you do not already have them, add PLT definitions for the PLTPI and PLTSD tables.

Program Entries

AVB requires entries for the following programs and tables in the CSD (formerly the PPT). These entries belong to the group AVBV5:

AVBCICS	AVBCICS5	AVBCIVP
AVBCICS0	AVBCICS6	AVBFIXTB
AVBCICS1	AVBCICS7	AVBOPT
AVBCICS2	AVBCICS8	AVBPROG
AVBCICS3	AVBCICS9	AVBTAB
AVBCICS4	AVBCICSV	AVBREQT

In addition, an entry in the same group is required for the command-level link routine. For this, use the name specified in the ENTPT parameter in ADAGSET; this is normally ADABAS.

An entry in the same group is also required for the task-related user exit. For this, use the name specified in the TRUENM parameter in ADAGSET; this is normally ADATRUE.

The following attribute settings apply to **all** AVB programs and tables:

Attribute	Value
Group	AVB51
Language	Assembler
Reload	No
Usage	Normal
Uselpacopy	No
Status	Enabled
Datalocation	Any
Execkey	Member-specific

Certain programs and tables **must** be defined as resident. The CEDF attribute is recommended for some programs but is not supported for others. The following table provides residence and CEDF specifications:

Module	Must Be Resident?	CEDF
AVBCICS	No	Optional
AVBCICS0	No	Recommended
AVBCICS1	No	Recommended
AVBCICS2	Yes	Optional
AVBCICS3	Yes	Not supported
AVBCICS4	Yes	Not supported
AVBCICS5	No	Optional
AVBCICS6	No	Recommended
AVBCICS7	No	Recommended
AVBCICS8	No	Recommended
AVBCICS9	No	Recommended
AVBCICSV	No	Optional
AVBCIVP	No	Recommended

Module	Must Be Resident?	CEDF
AVBOPT	Yes	Not supported
AVBPROG	Yes	Not supported
AVBTAB	Yes	Not supported
AVBFIXTB	No	Not supported
AVBREQT	Yes	Optional
ADABAS	Yes	Optional
ADATRUE	Yes	Optional

The CICS PLTPI and PLTSD tables also must be defined as resident.

Note:
When you need to load a new copy of a resident AVB module, use CICS online resource definition. Do ***not*** *use CEMT NEW; if you do, the program will be marked as nonresident, and processing results will be unpredictable.*

All other attribute settings are at the discretion of the site.

Transaction Entries

AVB version 5.1 requires entries for the following transactions in the CSD (formerly the PCT). These entries belong to the group AVB51.

AVB0	AVIP	AVC1
AVB1	AVIR	AVCI (optional; allows use of CECI to test AVB)
AVB5	AVTR	
AVB9	AVZP	

File Entries

AVB version 5.1 requires a CSD or FCT entry for each test file that you loaded into your environment. These entries belong to the group AVBFILES.

Note:
*Each file to be processed through AVB under CICS **must** be defined as CLOSED,UNENABLED. This definition ensures that the file is not opened before AVB is activated.*

- **COBOL**

SYS021	COBOL KSDS file
SYS0211	COBOL KSDS file; alternate index
SYS0212	COBOL KSDS file; alternate index
SYS0213	COBOL KSDS file; alternate index
SYS021R	COBOL RRDS file

.
.

- **PL/1**

PLITEST	PL/1 KSDS file
PLIAIX1	PL/1 KSDS file; alternate index
PLIAIX2	PL/1 KSDS file; alternate index
PLIESDS	PL/1 ESDS file

.
.
.

PLIRRDS	PL/1 RRDS file

.
.

Step 14. Install AVB Online Services

You can load the programs and messages for AVB online services using standard Natural INPL and ERRLODUS procedures:

- Use the Natural INPL utility to load the AVB online services programs into the Natural system file. The programs are loaded to the Natural application SYSAVB.
- Use the Natural ERRLODUS utility to load the AVB online services error messages into the Natural error library. Messages specific to the AVB online services application have numbers with the prefix “AVB”.

Refer to the *Natural Administration Manual* for information about using these utilities and their parameters.

To run the INPL and load the error text from the AVB distribution tape

Use the following sample JCS and check the reports produced to ensure that no errors have occurred.

Make the following changes for your site:

Change . . .	To . . .
adalib	the correct name for the Adabas PROC
cuu	the appropriate address for the tape unit
device	the device type of the external storage device on which the first block of the Adabas Associator is stored.
dbid	ID of the physical database where Adabas is installed
dddd	Id of the physical database where Natural is installed
fileno	file number for the Natural FUSER system file
fnr	file number for the Natural FNAT system file
natlib	the correct name for the Natural PROC
p	indicate the position of the file on the tape (see the *Report of Tape Creation*)
svc	SVC to be used for the current Adabas session
vrs	indicate the version, revision, and SM level, respectively

```
// JOB AVBINPL  LOAD THE AVB ONLINE SERVICES DATA INTO NATURAL
// OPTION LOG,NODUMP
* ******************************************************
*       LOAD NATURAL PROGRAMS FROM AVBvrs.INPL ON TAPE
* ******************************************************
// EXEC PROC=natlib
// EXEC PROC=adalib
// LIBDEF PHASE,SEARCH=(natlib.NATvrs,adalib.ADAvrs),TEMP
// TLBL CMWKF01,'AVBvrs.INPL',,,,p
// ASSGN SYS000,READER
// ASSGN SYS001,cuu
// EXEC NATBATWT,SIZE=NATBATWT,PARM='SYSRDR'
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=','
DBID=dddd,FNR=fnr,FUSER=(,fileno)
BWORKD=(1,1,4628,VB,2,2,4628,VB,3,3,4628,VB,4,4,4628,VB)
BWORKDL=(1,SL,2,SL,3,SL,4,SL)
STACK=INPL
/*
ADARUN DB=dbid,DEVICE=device,SVC=svc,MODE=MULTI
/*
B
FIN
/*
// MTC REW,SYS001
// ASSGN SYS001,UA
* ******************************************************
*       LOAD ERROR TEXTS FROM AVBvrs.ERRN ON TAPE
* ******************************************************
// EXEC PROC=natlib
// EXEC PROC=adalib
// LIBDEF PHASE,SEARCH=(natlib.NATvrs,adalib.ADAvrs),TEMP
// TLBL CMWKF02,'AVBvrs.ERRN',,,,p
// ASSGN SYS000,READER
// ASSGN SYS002,cuu
// EXEC NATBATWT,SIZE=NATBATWT,PARM='SYSRDR'
IM=D,MADIO=0,MT=0,OBJIN=R,AUTO=OFF,MAXCL=0,ID=','
DBID=dddd,FNR=fnr,FUSER=(,fileno)
BWORKD=(1,1,4628,VB,2,2,4628,VB),BWORKDL=(1,SL,2,SL)
/*
ADARUN DB=dbid,DEVICE=device,SVC=svc,MODE=MULTI
/*
LOGON SYSERR
ERRLODUS
FIN
/*
// MTC REW,SYS002
/*
```

Step 15. Verify the CICS Installation

AVB provides the AVBCIVP program to verify the installation under CICS/ESA. The AVCI transaction can be used for additional testing.

Using the AVBCIVP Program

To verify AVB under CICS using AVBCIVP

1. Define the AVBCIVP program and the AVIP transaction to CICS.
2. Modify member AVBCASM.X in the AVB library.
3. Supply job information in source member ASMCIVP.X.
4. Run ASMCIVP.X to preprocess, assemble, and link the IVP.
5. Execute the transaction AVIP. Information about the completed transaction is displayed on the screen. If AVB has been installed correctly under CICS, the bottom line of the screen should read as follows:

   AVIP – TRANSACTION ENDED NORMALLY

Using the AVCI Transaction

The AVCI transaction provides a quick and flexible way to test VSAM requests that will be handled by AVB. AVCI must be defined to CICS as an alias for the CICS-supplied CECI transaction.

To test the installation using AVCI

1. If AVB was not activated at CICS startup, execute the AVB1 transaction. Use the AVB1 commands ON and FILES / OPEN to activate AVB and open the IVP files.

   The password for the AVB1 transaction is the group password defined in the source member AVBTAB.

2. Use AVCI (an alias for CECI) to issue a START BROWSE command. The following example is for the COBOL file SYS021:

   AVCI STARTBR FI(SYS021) RI(&K)

   An error message is returned. Press PF5 and define the following variables:

   &K Specify the key length; this value must match the KEYLEN parameter in the transparency table. For the SYS021 test file, KEYLEN=7.

   &I Specify the record length; this value must match the RECSIZ parameter in the transparency table. For the SYS021 test file, RECSIZ=80.

   Press ENTER several times until the command is completed.

3. Enter the following command to perform a READ NEXT:

   READN FI(SYS021) RI(&K) INTO(&I) LENGTH(&L)

   Perform several READ NEXT commands.

4. Enter the following command to end the BROWSE:

   ENDBR FI(SYS021)

5. Start the transaction again and read using an alternate index:

   AVCI STARTBR FI(SYS021n) RI(&K)

   READN FI(SYS021n) RI(&K) INTO(&I) LENGTH(&L)

   —where “n” is 1, 2, or 3.

4

MIGRATING VSAM APPLICATIONS TO ADABAS

Operating Options

AVB may be executed in the following file environments:

- Adabas files only;
- VSAM files only;
- Both Adabas and VSAM files (mixed environments).

Note:
Currently, all Adabas files needed for a transaction or an application program must exist in the same physical database.

The ability to operate in a mixed environment means that your migration schedule can be tailored to your needs and resources. You can migrate files from VSAM to Adabas as needed, one application or even one file at a time.

Migration Procedure

Once AVB is installed, use the following procedure to migrate a VSAM application to Adabas:

Step 1. Define an Adabas file structure for each VSAM file

Define an Adabas file structure for each VSAM record referenced in the application program.

Design the Adabas file to correspond to the VSAM file while taking advantages of Adabas features. See the *Adabas DBA Reference Manual* for a discussion of file design. Create the Adabas field definition table (FDT).

The chapter **Defining the Adabas File Structure** starting on page 81 provides guidelines for defining the Adabas file structure, along with examples of how to support VSAM structures and attributes in Adabas.

Step 2. Unload the VSAM file(s) into a sequential dataset

Unload the VSAM file into a sequential dataset using the REPRO function of the IBM utility IDCAMS.

Step 3. Load the VSAM file(s) into an Adabas database

Load the data into the Adabas file using Adabas load utilities ADACMP and ADALOD.

Consult the *Adabas Utilities Manual* for information about these utilities.

Use the recommended naming conventions described on page 94.

Step 4. Prepare the transparency table

The AVB transparency table maps the old VSAM file structure to the new Adabas field definition table (FDT). The chapter **Creating the Transparency Table** starting on 95 page describes transparency-table entries.

1. For each file created in steps 1–3, create an entry in the transparency table to map the VSAM record structure to the new Adabas file structure.

   If the file is to be bridged under CICS, use RDO or some other CICS utility to create a CSD entry for the VSAM file.

2. Reassemble and relink the transparency table using the source members ASMTABB (for batch) and ASMTABC (for CICS).

   Note:
   Since AVB operates in a mixed environment, you may migrate VSAM files and applications to Adabas gradually. However, each time you change the transparency table, you must relink it for both batch and CICS processing.

Step 5. Modify the job stream

1. Include * AVBUSER and ADARUN parameter statements in the job stream.

2. For batch execution, modify the job control statements so that the application program invokes AVB.

   For execution under CICS, provide a load library containing the AVB load modules in the CICS DFHRPL concatenation. Define the AVB programs to CICS using RDO or the DFHCSDUP utility.

Step 6. (Optional; batch only) Add restart/recovery capability

Modify the application programs to add Adabas restart/recovery capability.

The chapter **Adding Restart/Recovery Support for Batch Programs** starting on page 217 discusses this procedure.

5

DEFINING THE ADABAS FILE STRUCTURE

This chapter provides guidelines for defining the Adabas file structure. The general definition procedure is followed by sections explaining how to define in Adabas

- VSAM keys;
- repeating fields and groups; and
- multiple record types in a file.

The naming conventions used in the examples for Adabas fields are listed on page 94. Software AG suggests that you use these conventions for your files as well.

General Procedure

To define an Adabas file for a VSAM file referenced by your application program

1. Design the Adabas file to correspond to the VSAM file while taking advantage of Adabas features.

   Consult the *Adabas DBA Reference Manual* for a discussion of file design.

2. Use the ADACMP utility or Adabas Online System (AOS) to create the Adabas field definition table (FDT) in Adabas.

   The definition of Adabas fields for VSAM fields is generally a straightforward process. However, to take advantage of Adabas features such as null suppression, you can define and load fields in an order different from that in the VSAM file. And in Adabas, field levels are increased by 1.

   Using ADACMP, the FDT is created from the field-definition statements when you compress the records before loading them into Adabas. For information about ADACMP, consult the *Adabas Utilities Manual.*

   Using AOS, the FDT is created with File Maintenance functions. You cannot define superdescriptors or subdescriptors using this method. For more information, consult the *Adabas DBA Reference Manual* (or the *Adabas Online System Manual* for Adabas version 7 and above).

The Adabas FDT can also be generated using Predict, Software AG's active data dictionary system, by defining the file in Predict as a type A object and generating the FDT and the ADACMP input statements from the Predict object. Consult the Predict documentation for more information.

Defining VSAM Keys

Define each VSAM key in Adabas as a **descriptor** field. A key can be defined in Adabas as a descriptor, subdescriptor, or superdescriptor.

Define the Adabas field with a length that will accommodate the largest value expected by the application program. Software AG recommends that this length be the same as the length used in the original VSAM file.

Define keys in Adabas with one of two data formats:

- packed decimal (P)

  The packed-decimal format can be used only under certain conditions, which are described in the section **Defining a Key in Packed-Decimal Format** on page 84.

- alphanumeric (A)

  All other VSAM keys must be defined as alphanumeric in Adabas, including keys that are a combination of packed decimal and another format. For "generic READs", the key must be alphanumeric.

Defining the Primary Key

Define the primary key as a **unique** descriptor in Adabas; otherwise, Adabas cannot detect a duplicate key.

Note:
*The key **must** also be defined as unique in the AVB transparency table (MCTAB parameter UNIQUE=Y). The UNIQUE parameter is described on page 105.*

Example 1: An Elementary Field as the Key

In the INVOICES file, the INV–NO field is the primary key:

COBOL Definition

```
01 INVOICE.
     05 INV–NO            PIC X(8).
```

Adabas Field Definition Table

INV–NO is defined in Adabas as the unique descriptor SK (DE,UQ).

COBOL Field Name	Level	Name	Length	Format	Options
01 INVOICE	01	GA			
05 INV–NO	02	SK	8	A	DE,UQ

Example 2: A Superdescriptor as the Key

In the STUDENTS file, the primary key is group KEY comprising the ID–NO and RECTYPE fields:

COBOL Definition

```
01 STUDENT–INFO.
     05 KEY.
        10 ID–NO            PIC X(6).
        10 RECTYPE          PIC X(1).
```

Adabas Field Definition Table

The KEY group is defined in Adabas as group KK, which includes member fields K1 and II.

COBOL Field Name	Level	Name	Length	Format	Options
01 STUDENTS	01	GA			
05 KEY	02	KK			
10 ID–NO	03	K1	6	A	UQ
10 RECTYPE	03	II	1	A	

However, in Adabas, the primary key is the unique superdescriptor SK, which is created from the K1 and II fields:

```
SUPDE='SK=K1,II,UQ'
```

Defining an Alternate Key

Alternate keys are defined in Adabas the same way that primary keys are defined, except that alternate keys do not have to be defined as unique. (However, define the field as unique if you want to prohibit duplicate values.)

Note:
When a secondary key is defined as unique in the transparency table (MCTAB keyword UNIQUE=Y), the Adabas field definition for the two-byte key field ***must*** *specify the UQ option. Adabas rather than AVB will then enforce the unique value for the key, ensuring better performance.*

Defining a Key in Packed-Decimal Format

The packed-decimal (P) format can be used only if **both** of the following conditions are satisfied:

- The entire key is a properly formed packed-decimal field or fields (containing digits and a sign);
- The packed-decimal field(s) is referenced in its entirety.

You can derive an Adabas superdescriptor from two packed-decimal fields when the fields are properly formed and the superdescriptor is composed of the entire fields.

Note:
You must also define a packed-decimal key to AVB using the transparency-table parameters PCKKEY, PKOFF1, and PKOFF2. These parameters are discussed on page 107.

Example

In the INVOICES file, CUST–NO and LNAME are alternate keys. CUST–NO is a a unique, packed field. LNAME is not unique.

COBOL Definition

```
01 INVOICE.
   05 INV–NO            PIC X(8).
   05 CUST–NO           PIC S9(7) COMP–3.
   05 FNAME             PIC X(20).
   05 LNAME             PIC X(25).
```

Adabas Field Definition Table

COBOL Field Name	Level	Name	Length	Format	Options
01 INVOICE	01	GA			
05 INV—NO	02	SK	8	A	DE,UQ
05 CUST—NO	02	A1	4	P	DE,UQ
05 FNAME	02	AA	20	A	
05 LNAME	02	A2	25	A	DE

Defining Repeating Fields and Groups

Using MUs and PEs

Use Adabas multiple-value fields (MUs) and periodic groups (PEs) to support VSAM repeating fields and groups:

- Define an Adabas MU for a single VSAM repeating field.
- Define an Adabas PE for a repeating VSAM group.

Note:
Always define MUs and PEs as level 1 fields and reference them explicitly in the Adabas format buffer. The format buffer is specified in the AVB transparency table.

Example

COBOL Definition

```
01 STUDENT—INFO.
   .
   .
   05 ADVISORS            PIC X(20) OCCURS 2 TIMES.
```

Adabas Field Definition Table

ADVISORS is defined in Adabas as MA, a level 1 MU with two occurrences.

COBOL Field Name	Level	Name	Length	Format	Options
05 ADVISORS	01	MA	20	A	MU(2

Unlike VSAM, Adabas permits fixed fields after an MU or PE.

An Adabas MU or PE can have up to 191 occurrences. The section **When the Number of Occurrences Exceeds Adabas Limits** on page 87 tells how to define an MU or PE for a VSAM repeating field or group that exceeds these limits.

When the Number of Occurrences Varies

For repeating fields or groups with variable numbers of occurrences, use a user exit 6 routine to add a one-byte occurrence count to each record before loading the file into Adabas. For more information, consult the discussion of ADACMP in the *Adabas Utilities Manual*.

Example

In the STUDENTS file, the PAYMENTS group occurs 1 to 3 times depending on the value in the PMTS field.

COBOL Definition

```
05 PMTS              PIC 9(1).
05 PAYMENTS OCCURS 1 TO 3 TIMES DEPENDING ON PMTS.
   10 DATE           PIC 9(6).
   10 AMOUNT         PIC 9(6).
   10 MEMO           PIC X(40).
```

Adabas Field Definition Table

PAYMENTS is defined in Adabas as P1. The number of occurrences varies and therefore is not specified in the FDT.

COBOL Field Name	Level	Name	Length	Format	Options
05 PMTS	02	O1	1	U	NU
05 PAYMENTS	01	P1			PE
10 DATE	02	AM	6	U	
10 AMOUNT	02	AN	6	U	
10 MEMO	02	AP	40	A	

Use an Adabas user exit 6 routine to build the Adabas record from the file created by the IDCAMS REPRO function.

The AVB source member COBUEX6 provides a model. COBUEX6 builds Adabas records for the KSDS test file used to verify the AVB installation; this file includes both OCCURS DEPENDING ON clauses and multiple record types.

When the Number of Occurrences Exceeds Adabas Limits

When a VSAM repeating field exceeds the number of occurrences allowed for Adabas MUs or PEs, define multiple MUs or PEs with 64 occurrences in each.

The number of occurrences in the last MU/PE is the remainder obtained after dividing the maximum number of occurrences by 64.

Example

In the INVOICES file, the ITEMS group occurs 1 to 150 times depending on the value in ITEM–CNT.

COBOL Definition

```
05 ITEM–CNT            PIC 9(3).
05 ITEMS OCCURS 1 TO 150 TIMES DEPENDING ON ITEM–CNT.
   10 ITEM–NO          PIC 9(5)
   10 ITEM–QTY         PIC 9(3).
   10 ITEM–PRC         PIC 9(6).
```

Adabas Field Definition Table

The site is running Adabas version 5.2, which limits PEs to 99 occurrences. Because the maximum number of ITEMS occurrences exceeds the limit for a PE, three PEs must be defined.

COBOL Field Name	Level	Name	Length	Format	Options
05 ITEM–CNT	02	01	3	U	
05 ITEMS	01	P1			PE(64)
10 ITEM–NO	02	AB	5	U	NU
10 ITEM–QTY	02	AC	3	U	NU
10 ITEM–PRC	02	AD	6	U	NU

COBOL Field Name	Level	Name	Length	Format	Options
05 ITEMS	01	P2			PE(64)
10 ITEM–NO	02	AE	5	U	NU
10 ITEM–QTY	02	AF	3	U	NU
10 ITEM–PRC	02	AG	6	U	NU
05 ITEMS	01	P3			PE(22)
10 ITEM–NO	02	AH	5	U	NU
10 ITEM–QTY	02	AJ	5	U	NU
10 ITEM–PRC	02	AK	6	U	NU

The PE occurrences correspond to those in the VSAM file as follows:

Adabas PE	VSAM Occurrences
P1	1–64
P2	65–128
P3	129–150

Restrictions

- Position is not maintained for a null value in an occurrence of a null-suppressed MU. If your program logic requires specific occurrences of a repeating field, do **not** specify null suppression for the corresponding MU field.
- In COBOL, all PEs must be defined as alphanumeric.
- Adabas syntax does not permit an MU within a range of PE occurrences. Thus, AVB does **not** support MUs nested within a PE for OCCURS DEPENDING ON processing, which uses ranges of occurrences.

Defining Multiple Record Types

Sequence for Field Definitions

When a file has multiple record types, it may be useful to define the fields that are common to all record types first in the FDT. Then define the fields that belong to different record types and define them with null suppression where possible.

Use an Adabas user exit 6 routine to build the Adabas record from the file created by the IDCAMS REPRO function. The AVB source member COBUEX6 provides a model.

Note:
Several transparency-table parameters are used to define multiple-record-type files to AVB. These parameters are described on page 120.

Example

The STUDENTS file contains two record types, STUDENT–INFO and STUDENT–ACCT. The RECTYPE field indicates the type of record.

COBOL Definition

```
FD STUDENTS
   LABEL RECORDS ARE STANDARD
   DATA RECORDS ARE STUDENT—INFO STUDENT—ACCT

01 STUDENT—INFO.                                  [FIRST RECORD TYPE]
   05 KEY.
      10 ID—NO          PIC X(6).
      10 RECTYPE        PIC X(1).
   05 FULL—NAME.
      10 FNAME          PIC X(15).
      10 LNAME          PIC X(20).
   05 ADDRESS.
      10 STREET         PIC X(25).
      10 CITY           PIC X(20).
      10 STATE          PIC X(2).
      10 ZIP            PIC 9(9).
   05 DEPT              PIC X(4).
   05 ADVISORS          PIC X(20) OCCURS 2 TIMES.
```

```
01 STUDENT-ACCT.                                    [SECOND RECORD TYPE]
   05 KEY.
      10 ID-NO          PIC X(6).
      10 RECTYPE        PIC X(1).
   05 TERM              PIC X(6).
   05 CHARGES.
      10 TUITION        PIC 9(6).
      10 ROOM           PIC 9(5).
      10 MEALS          PIC 9(5).
   05 PMTS              PIC 9(1).
   05 PAYMENTS OCCURS 1 TO 3 TIMES DEPENDING ON PMTS.
      10 DATE           PIC 9(6).
      10 AMOUNT         PIC 9(6).
      10 MEMO           PIC X(40).
```

Adabas Field Definition Table

The KEY group is common to both record types. This group is defined first in Adabas, followed by the fields specific to each record type. Note that the MU and PE are defined with level 1.

COBOL Field Name	Level	Name	Length	Format	Options
[COMMON FIELDS]	01	GA			
05 KEY	02	KK	6	A	UQ
10 ID-NO	03	K1	6	A	UQ
10 RECTYPE	03	II	1	A	
[FIRST RECORD TYPE]	01	GB			
05 FULL-NAME	02	GC			
10 FNAME	03	AA	15	A	NU
10 LNAME	03	AB	20	A	NU
05 ADDRESS	02	GD	25	A	NU
10 STREET	03	AC	25	A	NU
10 CITY	03	AD	20	A	NU
10 STATE	03	AE	2	A	NU
10 ZIP	03	AF	9	U	DE
05 DEPT	02	AG	4	A	
05 ADVISORS	01	MA	20	A	MU(2)

COBOL Field Name	Level	Name	Length	Format	Options
[SECOND RECORD TYPE]	01	GE			
05 TERM	02	AH	6	A	NU
05 CHARGES	02	GF			
10 TUITION	03	AJ	6	U	NU
10 ROOM	03	AK	5	U	NU
10 MEALS	03	AL	5	U	NU
05 PMTS	02	01	1	U	NU
05 PAYMENTS	01	P1			PE
10 DATE	02	AM	6	U	
10 AMOUNT	02	AN	6	U	
10 MEMO	02	AP	40	A	

Restrictions

- Do not use an MU or PE for the record-type field.
- Do not specify null suppression for the record-type field. Adabas returns no record when a null-suppressed field is empty.
- When a field in an application is redefined and one of the redefinitions specifies character format, define the Adabas field as alphanumeric.

Completed Examples

COBOL Field Name	Level	Name	Length	Format	Options
01 INVOICE	01	GA			
05 INV–NO	02	SK	8	A	DE,UQ
05 CUST–NO	02	A1	4	P	DE,UQ
05 FNAME	02	AA	20	A	
05 LNAME	02	A2	25	A	DE
05 ITEM–CNT	02	01	3	U	
05 ITEMS	01	P1			PE(64)
10 ITEM–NO	02	AB	5	U	NU
10 ITEM–QTY	02	AC	3	U	NU
10 ITEM–PRC	02	AD	6	U	NU
05 ITEMS	01	P2			PE(64)
10 ITEM–NO	02	AE	5	U	NU
10 ITEM–QTY	02	AF	3	U	NU
10 ITEM–PRC	02	AG	6	U	NU
05 ITEMS	01	P3			PE(22)
10 ITEM–NO	02	AH	5	U	NU
10 ITEM–QTY	02	AJ	3	U	NU
10 ITEM–PRC	02	AK	6	U	NU
01 STUDENTS	01	GA			
05 KEY	02	KK			
10 ID–NO	03	K1	6	A	UQ
10 RECTYPE	03	II	1	A	
	01	GB			
05 FULL–NAME	02	GC			
10 FNAME	03	AA	15	A	NU

COBOL Field Name	Level	Name	Length	Format	Options
10 LNAME	03	AB	20	A	NU
05 ADDRESS	02	GD			
10 STREET	03	AC	25	A	NU
10 CITY	03	AD	20	A	NU
10 STATE	03	AE	2	A	NU
10 ZIP	03	AF	9	U	DE
05 DEPT	02	AG	4	A	
05 ADVISORS	01	MA	20	A	MU(2)
	01	GE			
05 TERM	02	AH	6	A	NU
05 CHARGES	02	GF			
10 TUITION	03	AJ	6	U	NU
10 ROOM	03	AK	5	U	NU
10 MEALS	03	AL	5	U	NU
05 PMTS	02	01	1	U	NU
05 PAYMENTS	01	P1			PE
10 DATE	02	AM	6	U	
10 AMOUNT	02	AN	6	U	
10 MEMO	02	AP	40	A	

Naming Conventions for Adabas Fields

Software AG recommends that you use the following field naming conventions for the Adabas files you define:

Field	Description
SK	Primary key, which must be a unique descriptor, super- or subdescriptor. Any name may be used for the primary key, but "SK" is recommended.
KK, K1–9	Parent field or fields when SK is a super- or subdescriptor. The II field is also a parent field.
II	Indicator field. Used for record-type field in multiple-record-type files.
A1–9	Alternate keys. Although these names are recommended, the names or quantity of alternate keys are not restricted.
GA–Z	(01) Group fields.
MA–Z	Multiple-value fields (MUs).
M1–9	MUs that correspond to COBOL fields defined by OCCURS DEPENDING ON clauses.
L1–9	Fields (the objects of OCCURS DEPENDING ON clauses) that indicate the number of occurrences in fields M1–M9.
PA–Z	Periodic groups (PEs).
P1–9	PEs that correspond to COBOL fields defined by OCCURS DEPENDING ON clauses.
O1–9	Fields (the objects of OCCURS DEPENDING ON clauses) that indicate the number of occurrences in fields P1–9.

Notes:

1. *MUs should be defined as 01 levels. Rules for coding MUs must be followed as documented in the **Adabas Command Reference Manual**.*
2. *All PEs must be defined with alphanumeric format.*

6

CREATING THE TRANSPARENCY TABLE

The AVB transparency table maps VSAM files referenced in application programs to corresponding Adabas files. The table is created by the Assembler macro MCTAB. MCTAB is called automatically by the job ASMTABB or ASMTABC, which assemble and link the transparency table for batch or CICS processing, respectively.

You can create new MCTAB statements to be used by these jobs, or you can copy and modify the statements in the source member AVBTAB (or IVPTAB for VSE/ESA) that was used for the test files and programs.

This chapter describes the four types of MCTAB macro statements and the parameters used in these statements. The last section provides instructions for assembling and linking the transparency table.

Note:
Each time you change the transparency table, you must reassemble and relink it using one or both of the jobs ASMTABB and ASMTABC.

Alternatively, the AVB transparency table can be generated using Predict, Software AG's active data dictionary system, although some optional parameters are not supported. Consult the Predict documentation for more information.

To convert from an earlier version of AVB

1. Remove the following parameters from your transparency table(s); they are no longer used: KEYB, ODFMT, ODLEN, ODOFF, ODPOS, OSOFF, OSSIZ, PREFNUM.
2. Add the following new parameters to your transparency table(s), if required: FDSN (page 103), LALEN and LANAME (page 119), ODMAX (page 114), and PKOFF3 (page 107).
3. Modify the relevant format buffer specification if you are using repeating fields.
4. Reassemble and relink your transparency table(s) using the AVB version 5.1 source library and macros.

MCTAB Statements

The macro MCTAB uses the following four types of statements:

Type	Function
INITIAL	Creates a dummy entry that signals the start of the table. There must be one and only one INITIAL statement.
END	Creates a dummy entry that signals the end of the table. There must be one and only one END statement.
ETOPNL	Used for batch applications only; specifies a list of files for which the batch AVBETBT module is to read and write ET data or for which the Adabas OP command is to retrieve ET data when the files are opened. This statement is optional. If it is specified, there must be only one; it must follow the END statement and be the last entry in the transparency table..
GEN	(the default) Creates an entry for a file in the transparency table. There must be one GEN entry for each file in the transparency table, and one for each alternate (secondary) index.

Parameter Syntax

Since the AVB transparency table is created by the Assembler macro MCTAB, parameter syntax conforms with Assembler syntax. In particular, note the following requirements:

- Within an entry, use commas to separate parameters.
- When multiple values are specified for a parameter, enclose the values in parentheses and separate them with commas; for example, ODNAME=(M1,P1,M2).
- To continue an entry to another line, enter a nonblank continuation character in column 72, and continue in column 16 of the next line.
- Embedded blanks are not allowed, except when an entry is continued to another line; in that case, blanks may precede the continuation character in column 72.
- Enclose specifications for Adabas format buffers (in the FORMAT or RECFMTS parameters) in apostrophes.

Parameter Overview

The following MCTAB parameters are described in this chapter on the page specified:

Parameter	Statement	Description	Page
DBID	INITIAL	the global Adabas DBID for the AVB session.	100
DBID	GEN	an Adabas DBID for the session; used to override the DBID specified on the INITIAL statement for a specific file.	100 128
FDSN	GEN	used to specify an OS/390 DSName or VSE DLBL dataset name for the VSAM file in AVB that is different from the VSAM file name outside of AVB.	103
FN	GEN	the VSAM file name.	103
FNR	GEN	the Adabas file number.	103
FORMAT	GEN	the format buffer that AVB passes to Adabas up to 255 bytes.	109
FORMATX	GEN	the format buffer that AVB passes to Adabas beyond 255 bytes.	109
GLBLFID	GEN	a global format ID for the Adabas format buffer specified by the FORMAT parameter.	111
GPWD	INITIAL	the password for the CICS transaction AVB1.	101
INDXTYP	GEN	whether KEY1 is a primary or secondary key.	105
KEY1	GEN	the name of the descriptor, subdescriptor, or superdescriptor used for the primary or secondary key.	104
KEYLEN	GEN	the length of the key field.	104
KEYOFF	GEN	the offset of the key from the beginning of the record.	105
LALEN	GEN	the fixed length (in bytes) of the long-alpha fields.	119
LANAME	GEN	the Adabas names of the long-alpha fields.	119
OCIND	GEN	whether the file will be open or closed when starting AVB in a CICS environment or whether the file will be bypassed by AVB processing and opened by native VSAM.	129

Parameter	Statement	Description	Page
ODINDX	GEN	the Adabas names used for the occurs-depending-on (ODO) index fields.	114
ODMAX	GEN	the maximum number of occurrences for the occurs-depending-on (ODO) repeating fields.	114
ODNAME	GEN	the Adabas names of the MU or PE fields used for the occurs-depending-on (ODO) repeating fields.	114
PCKKEY	GEN	whether the format of the key specified in the GEN entry is packed.	107
PKOFF1	GEN	the offset of the sign from the first byte of the packed field.	108
PKOFF2	GEN	the offset of the sign from the first byte of the second packed field.	108
PKOFF3	GEN	the offset of the sign from the first byte of the third packed field.	108
PREFSZE	GEN	the size (in kilobytes) of the ISN buffer to be allocated for AVB record prefetching.	126
PWD	GEN	the password to be inserted in Adabas calls for access to the file.	128
RECFMTS	GEN	the format buffer for each record type up to 255 bytes as required.	123
RECFMTX	GEN	the format buffer for each record type beyond 255 bytes up to 510 bytes as required.	123
RECFMTY	GEN	the format buffer for each record type beyond 510 bytes up to 765 bytes as required.	123
RECFMTZ	GEN	the format buffer for each record type beyond 765 bytes up to 1020 bytes as required.	123
RECIDFB	GEN	the name of the Adabas record-type field.	122
RECSIZ	GEN	the length of the VSAM record.	103
RECTBYT	GEN	the length (in bytes) of the record-type field in Adabas.	122
RECTCNT	GEN	the number of record types in the file.	121
RECTFMT	GEN	whether the record-type indicator specified in the RECFMTS parameter is alphanumeric or numeric.	123

Parameter	Statement	Description	Page
RECTOFF	GEN	the offset (in bytes) from the beginning of the record to the record-type field.	123
RECTYPE	GEN	whether the file has more than one record type.	121
RESET	GEN	how AVB will handle a request to open a file when the RESET ACB indicator is set.	129
UNIQUE	GEN	whether values in the key field specified in the GEN entry must be unique.	105
UPD	ETOPNL	list of each Adabas file for which ET data is to be read or written up to 256 bytes.	102
UPD2	ETOPNL	continues UPD file list beyond 256 bytes.	102
VARLRECL	GEN	whether the file will support VSAM variable records.	103
VSAMTYP	GEN	whether the VSAM file is ESDS or RRDS.	104

The INITIAL Statement

The INITIAL statement creates a dummy entry that signals the start of the transparency table. There must be **one and only one** INITIAL statement.

The INITIAL statement has two parameters DBID and GPWD, both of which are optional.

Parameter	Specify . . .	Minimum	Maximum	Default
DBID	the global Adabas DBID for the AVB session.	0	65535	0

This parameter specifies the global Adabas DBID for the session. The default value (0) indicates that the DBID specified in the Adabas link routine is used as the global DBID.

AVB inserts the value specified by this parameter into the Adabas control block; thus it is used for every file accessed during the Adabas session.

You can override the global DBID for a specific file by specifying the DBID parameter in the GEN entry for the file.

For more information about how AVB determines the Adabas DBID to be inserted into the Adabas control block, consult the section **Overriding the Global Adabas DBID** on page 128. The maximum DBID is 65535.

Note:
AVB requires that all Adabas files used by a transaction or application program exist in the same physical database.

Parameter	Specify . . .	Possible Values	Default
GPWD	the password for the CICS transaction AVB1.	password	AVBRIDGE

This parameter specifies the password for the CICS transaction AVB1. AVB1 can be used to activate or deactivate AVB and to open or close files.

Example:

```
MCTAB TYPE=INITIAL,DBID=100,GPWD=AVB
```

The END Statement

The END statement creates a dummy entry that signals the end of the transparency table. There must be **one and only one** END statement.

There are no parameters for the END statement.

Example:

```
[GEN entries]
.
.
MCTAB TYPE=END
```

The ETOPNL Statement

The ETOPNL statement lists the files for which the batch AVBETBT module is to read and write ET data. It is required **only** when you use the AVBETBT module. If the ETOPNL statement is included, there must be **only one**; it must be the **last** entry in the table (after the END statement).

The ETOPNL statement has two parameters UPD and UPD2:

Parameter	Specify . . .	Possible Values	Default
UPD	file list of each Adabas file for which ET data is to be read or written up to 256 bytes.	filenumber	none
UPD2	file list of each Adabas file for which ET data is to be read or written beyond 256 bytes.		

The format of the file list used with the UPD and UPD2 parameters is

UPD='n,n,n,...'
UPD2='n,n,n,...'

The file list in UPD cannot exceed 256 bytes in length; if more than 256 bytes are needed, use the UPD2 parameter to continue a file list that was begun in UPD.

Use commas with no intervening blanks to separate Adabas file numbers; leading zeros are permitted. You do not need to terminate the list with a period; the MCTAB macro supplies one. An embedded period terminates the list.

Use a non-blank continuation character in column 72 to extend the list to multiple lines.

The specified file list is used when AVB issues an OP command.

Example:

```
[GEN entries]
.
.
MCTAB TYPE=END
MCTAB TYPE=ETOPNL,UPD='11,15,18'
```

The chapter **Adding Restart/Recovery Support for Batch Programs** starting on page 217 describes how to use the AVBETBT module to read and write batch ET data. For more information about constructing Adabas file lists, consult the *Adabas Command Reference Manual*.

Creating File Entries (GEN Statements)

AVB requires a GEN entry for each file and each alternate index to be processed through AVB. The following sections tell you how to define characteristics of files to AVB.

Defining a File to AVB

The following parameters define a file to AVB:

Parameter	Specify . . .	Possible Values	Default
FDSN	an optional parameter used for documentation purposes only that specifies a dataset name (DSN; DLBL for VSE/ESA) for the VSAM file in AVB that is different from the VSAM file name outside of AVB. Depending on the operating system, the name can be 1–44 alphanumeric characters beginning with an alpha character.	datasetname	none
FN	the VSAM file name. For batch, use the DD name (the DTF name for VSE/ESA); for CICS, use the FCT file name. The name can be 1–8 characters that conform to the requirements set by the operating system.	filename	none
FNR	the Adabas file number.	0–5000	none
RECSIZ	the length of the VSAM record. If the records in the file are of variable length, the value of this parameter must be large enough to handle the maximum record length.	n	none

If you have set the options table (AVBOPT) parameter VARLRECL=ON, you must add the MCTAB parameter VARLRECL=ON to each file that will have support for VSAM variable records:

Parameter	Specify . . .	Possible Values	Default
VARLRECL	whether the file will support VSAM variable records.	OFF | ON	OFF

If the file is ESDS or RRDS, a fourth parameter is required to specify which type:

Parameter	Specify . . .	Possible Values	Default
VSAMTYP	whether the file is ESDS or RRDS.	ESDS | RRDS	none

If no value for VSAMTYP is specified, then KSDS file type is assumed.

ESDS or RRDS dataset types also require PREFSZE=0. See page 126.

Examples:

```
MCTAB TYPE=GEN,FN=INVOICES,FNR=15,RECSIZ=2161

MCTAB TYPE=GEN,FN=DEMO,FNR=21,RECSIZ=34,VSAMTYP=RRDS
```

Defining the VSAM Keys to AVB

To improve performance, the Adabas field corresponding to the VSAM primary key must be defined in the Adabas field definition table (FDT) as a unique descriptor.

When a primary key is defined, or a secondary key is defined as unique (MCTAB keyword UNIQUE=Y), the Adabas field definition for the two-byte key field must specify the UQ option.

Adabas rather than AVB will then enforce the unique value for the key, ensuring better performance.

Five parameters define a key field to AVB:

Parameter	Specify . . .	Possible Values	Default
KEY1	the name of the descriptor, subdescriptor, or superdescriptor used for the primary or secondary key. "SK" is the recommended name for the primary key.	two-character Adabas name	none
KEYLEN	the length of the key field. The value must match the length specified in the Adabas FDT.	value specified in the Adabas FDT	none

Parameter	Specify . . .	Possible Values	Default
KEYOFF	the offset of the key from the beginning of the record.	offset	0
INDXTYP	whether the key (KEY1) is a primary (P) or secondary (S) key.	P | S	P
UNIQUE	whether values in the key field specified in the GEN entry must be unique. A key is defined to AVB as unique unless you explicitly code N for this parameter. Primary keys must be unique (the default).	N | Y	Y

Defining the Primary Key

To define the primary VSAM key to Adabas

1. Specify the KEY1, KEYLEN, and KEYOFF parameters.
2. Use the default value INDXTYP=P.
3. Use the default value UNIQUE=Y.

Note:
When a primary key is defined, the Adabas field definition for the two-byte key field ***must*** *specify the UQ option. Adabas rather than AVB will then enforce the unique value for the key, ensuring better performance.*

Example 1 : An Elementary Field as the Key

COBOL Field Name	Level	Name	Length	Format	Options
01 INVOICE	01	GA			
05 INV—NO	02	SK	8	A	DE,UQ

GEN Entry:

```
MCTAB TYPE=GEN,FN=EMPLREC,FNR=15,RECSIZ=2161,                    X
               KEY1=SK,KEYLEN=8,KEYOFF=0,INDXTYP=P
```

Example 2 : A Superdescriptor as the Key

COBOL Field Name	Level	Name	Length	Format	Options
01 STUDENTS	01	GA			
05 KEY	02	KK			
10 ID—NO	03	K1	6	A	UQ
10 RECTYPE	03	II	1	A	

Adabas Superdescriptor Definition:

```
SUPDE='SK=K1,II,UQ'
```

GEN Entry:

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                          X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P
```

Defining an Alternate (Secondary) Key

To define alternate VSAM keys to Adabas

1. Create a separate GEN entry for each alternate key for a file.

   For example, if you specify a primary key and three alternate keys in the same COBOL SELECT clause, create four GEN entries, one for the primary key and one for each of the alternate keys.

2. Specify a **different** VSAM file name (FN) in the entries for the primary key and each alternate key but the **same** Adabas file number (FNR) in all entries.

   If the entries are for a file bridged under CICS, ensure that there is an FCT or CSD entry for each file name.

3. Specify the KEY1, KEYLEN, and KEYOFF parameters for each alternate key.

4. Specify INDXTYP=S for each alternate key.

5. If an alternate key is **not** a unique descriptor, specify UNIQUE=N.

   Note:
   *When a secondary key is defined as unique (MCTAB keyword `UNIQUE=Y`), the Adabas field definition for the two-byte key field **must** specify the UQ option. Adabas rather than AVB will then enforce the unique value for the key, ensuring better performance.*

Example:

The VSAM primary key is INV–NO; CUST–NO and LNAME are alternate keys.

COBOL Field Name	Level	Name	Length	Format	Options
01 INVOICE	01	GA			
05 INV–NO	02	SK	8	A	DE,UQ
05 CUST–NO	02	A1	4	P	DE,UQ
05 FNAME	02	AA	20	A	
05 LNAME	02	A2	25	A	DE

GEN Entries:

- The primary key SK is defined in the entry for FN=INVOICES (the true VSAM file name).
- The alternate keys A1 and A2 are defined in the entries for FN=INVCUST and FN=INVNAME, respectively.
- FNR=15 appears in all three entries.

```
MCTAB TYPE=GEN,FN=INVOICES,FNR=15,RECSIZ=2161,                          X
               KEY1=SK,KEYLEN=08,KEYOFF=0,INDXTYP=P
MCTAB TYPE=GEN,FN=INVCUST,FNR=15,RECSIZ=2161,                          X
               KEY1=A1,KEYLEN=04,KEYOFF=08,INDXTYP=S
MCTAB TYPE=GEN,FN=INVNAME,FNR=15,RECSIZ=2161,                          X
               KEY1=A2,KEYLEN=25,KEYOFF=32,INDXTYP=S,UNIQUE=N
```

Defining a Packed-Decimal Key

The following parameters define a packed-decimal key:

Parameter	Specify . . .	Possible Values	Default
PCKKEY	whether the format of the key specified in the GEN entry is packed. If the key is a superdescriptor composed of – only packed-decimal fields, specify ‘Y’. – both packed-decimal and alphanumeric fields, use the default (‘N’).	N | Y	N

Parameter	Specify . . .	Possible Values	Default
PKOFF1	the offset of the sign from the first byte of the packed field. The sign is located in the last byte, so the offset value must be one less than the key length **except** for one-byte fields (KEYLEN=1); in that case, specify PKOFF1=1.	offset	0
PKOFF2	the offset of the sign from the first byte of the second packed field. See PKOFF1 above for more information. Use this parameter **only** for a superdescriptor composed of two packed keys.	offset	0
PKOFF3	the offset of the sign from the first byte of the third packed field. See PKOFF1 above for more information. Use this parameter **only** for a superdescriptor composed of three packed keys.	offset	0

Example:

The alternate key CUST–NO is a packed field.

COBOL Field Name	Level	Name	Length	Format	Options
01 INVOICE	01	GA			
05 INV–NO	02	SK	5	A	DE,UQ
05 CUST–NO	02	A1	4	P	DE,UQ
05 FNAME	02	AA	20	A	
05 LNAME	02	A2	25	A	DE

GEN Entry:

```
MCTAB TYPE=GEN,FN=INVCUST,FNR=15,RECSIZ=2161,                          X
               KEY1=A1,KEYLEN=04,KEYOFF=05,INDXTYP=S,PCKKEY=Y,PKOFF1=03
```

Specifying the Adabas Format Buffer

The Adabas format buffer determines how Adabas returns records to the application program. For the syntax of the format buffer, consult the *Adabas Command Reference Manual*.

Use the following parameters to specify the buffer that AVB passes to Adabas:

Parameter	Specify . . .	Possible Values	Default
FORMAT	the buffer that AVB passes to Adabas up to 255 bytes.	Adabas fieldname	none
FORMATX	the buffer that AVB passes to Adabas beyond 255 bytes.		

The values for the FORMAT and FORMATX parameters are specified in the following format:

FORMAT='xx[,xx,xx,xx,...]'

FORMATX='xx[,xx,xx,xx,...]'

—where "xx" represents an Adabas field name. Enclose specifications for format buffers in apostrophes. Use a comma to separate fields.

The length of the FORMAT parameter value cannot exceed 255 bytes. If the format buffer is longer than that, split the specification into two parts and use the FORMATX parameter in addition to FORMAT. Specify the first part of the buffer (up to 255 bytes) as the FORMAT value and the remainder as the FORMATX value.

Note:
For files with types, use the RECFMTS parameter instead of FORMAT. RECFMTS is described in the section ***Defining Multiple Record Types*** *on page 120.*

The Adabas format buffer **must** match the record format that the program expects. If this parameter is incorrect, Adabas returns a nonzero response code.

Example:

The COBOL file definition for a record in the EMPLOYEES file is shown below:

```
01 EMPL-REC.
   05 EMPL-NAME             PIC X(32).
   05 EMPL-SSNO             PIC 9(09).
   05 EMPL-DEPT             PIC X(05).
   05 EMPL-TITLE            PIC X(15).
   05 EMPL-PHONE            PIC X(04).
```

To take advantage of null suppression, the fields are defined to Adabas in a different order:

COBOL Field Name	Level	Name	Length	Format	Options
01 EMPL–REC	01	GA			
05 EMPL–SSNO	02	SK	9	A	DE,UQ
05 EMPL–NAME	02	NA	32	A	NU
05 EMPL–DEPT	02	DP	5	A	NU
05 EMPL–TITLE	02	TI	15	A	NU
05 EMPL–PHONE	02	TP	4	A	NU

GEN Entry:

The FORMAT value matches the field order in the COBOL definition.

```
MCTAB TYPE=GEN,FN=EMPLOYEE,FNR=11,RECSIZ=65,                          X
               KEY1=SK,KEYLEN=9,KEYOFF=0,INDXTYP=P,                   X
               FORMAT='NA,SK,DP,TI,TB'
```

In this example, FORMAT references each field separately because the COBOL program and Adabas FDT define the fields in a different order. If they defined fields in the same order, the group name could be used to reference all fields:

FORMAT='GA'

Using Global Format IDs

Software AG strongly recommends the use of Adabas global format IDs for online applications that share a common record layout. Global format IDs improve performance because Adabas does not have to rebuild the format buffer for each Adabas call.

Parameter	Specify . . .	Possible Values	Default
GLBLFID	a global format ID for the Adabas format buffer specified by the FORMAT parameter.	1–7 alphanumeric characters | DYNAMIC	none

AVB uses the GLBLFID value to build the global format ID. The global format ID has 8 bytes; the first byte is always X‘C0’. Bytes 2–8 are taken from the GLBLFID value for the format buffer used by the current Adabas command.

The GLBLFID value can be 1–7 alphanumeric characters; it must begin with a letter (A–Z). Enclose the value in apostrophes. The GLBLFID value can also be set to the constant DYNAMIC, which is **not** enclosed in apostrophes.

If the value DYNAMIC is used, AVB dynamically allocates a global format ID for the format buffer. This setting provides support for multiple-record type files and for files with OCCURS DEPENDING ON.

GLBLFID=DYNAMIC is now the recommended way to support global format IDs. See also the AVBOPT parameter DYNMGFID on page 137.

Example:

```
MCTAB TYPE=GEN,FN=EMPLOYEE,FNR=11,RECSIZ=65,                          X
               KEY1=SK,KEYLEN=9,KEYOFF=0,INDXTYP=P,                   X
               FORMAT='NA,SK,DP,TI,TB',GLBLFID=DYNAMIC
```

There must be a one-to-one correspondence between a GLBLFID value and a FORMAT value. AVB does not support global format IDs for multiple-record-type files unless the constant DYNAMIC is used instead of a user-generated value.

For information about global format IDs, consult the *Adabas Command Reference Manual*.

Defining Repeating Fields

In COBOL, the OCCURS clause specifies tables whose elements can be referred to by indexing or subscripting. The OCCURS clause can specify fixed-length or variable-length tables.

Variable-length tables are specified using the OCCURS DEPENDING ON (ODO) clause.

Example:

```
WORKING–STORAGE SECTION
01  TABLE–1.
    05  X                   PICS9.
    05  R OCCURS n TIMES
          DEPENDING ON X    PIC X.
```

“X” is the object and “R” is the subject of the OCCURS DEPENDING ON clause. In the following discussion, the ODO object “X” is called the “ODO index” and the ODO subject “R” is called the “ODO repeating field”. The value of the ODO index determines the number of ODO repeating fields to appear in the table. n specifies the maximum number of occurrences.

Fixed-Length Tables

When the number of occurrences does not vary; that is, for an OCCURS clause without DEPENDING ON, the repeating field must be specified in the FORMAT parameter using the syntax

FORMAT=’*...,xx001–nnn,...*’

—where

xx is the Adabas name of the MU or PE field corresponding to the repeating field; and

nnn is the number of occurrences.

- Ensure that the numbers ‘001’ and ‘nnn’ are always specified with three digits to simplify the AVB format buffer analysis.
- If the number of occurrences exceeds the Adabas limits, divide the repeating field over several Adabas MU/PE fields.

Note:
Do not specify any of the ODO GEN parameters described on page 114 in the transparency table for a fixed-length table. This was required for previous AVB versions, but is no longer allowed.

Example:

The repeating field ADVISORS has a fixed number of occurrences.

```
...
05 ADVISORS          PIC X(20) OCCURS 2 TIMES.
...
```

Adabas Field Definition Table:

COBOL Field Name	Level	Name	Length	Format	Options
05 ADVISORS	01	MA	20	A	MU(2)

GEN Entry:

Because the ADVISORS is a fixed-length table, none of the OD* parameters is required.

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,          X
               FORMAT='KK,GC,GD,AG,MA001–002'
```

Variable-Length Tables

When the number of occurrences varies; that is, for an OCCURS DEPENDING ON (ODO) clause, the repeating field must be specified in the FORMAT parameter using the syntax

FORMAT='...,*xx*001–N,...'

—where "xx" is the Adabas name of the MU or PE field corresponding to the repeating field.

Each of the MU/PE fields can contain at most 64 occurrences. When the number of occurrences of an ODO clause exceeds 64, use multiple MUs or PEs to define the VSAM repeating field:

FORMAT='*xx*001–N,[*xy*001–N,...,]*xz*001–N'

—where "xx", "xy", and "x*z*" are the names of MU/PE fields.

You must define enough MU/PE fields to cover the maximum number of occurrences:

- for 1 – 64 occurrences, define 1 PE/MU field

- for 65 – 128 occurrences, define 2 PE/MU fields
- for 129 – 192 occurrences, define 3 PE/MU fields; and so on..

For historical reasons, you may also use the syntax

FORMAT='....,xx001–nnn,...'

—where "nnn" is any three digit number. AVB ignores the value specified for "nnn".

Ensure that the number "001" is always specified with three digits to simplify the AVB format buffer analysis.

For READ requests, AVB uses the "–N" notation and for INSERT or UPDATE requests, it uses the value derived from the ODMAX parameter described in the following paragraphs.

AVB uses the following GEN parameters in the transparency table for the ODO clause:

Parameter	Specify ...	Value Format	Default
ODINDX	the two-character Adabas name(s) used for the ODO index field(s).	xx | (xx,xy,xz,...)	none
ODMAX	the maximum number of occurrences for the ODO repeating field(s).	n | (n,n,n,...)	none
ODNAME	the two-character Adabas name(s) of the MU(s) or PE(s) used for the ODO repeating field(s).	xx | (xx,xy,xz,...)	none

The ODO index field must be defined in Adabas with the format 'B' (binary), 'P' (packed), or format 'U' (unpacked). A binary field can be two or four bytes long; a packed field can be as many as 8 bytes long.

The ODNAME parameter must contain the first MU/PE field belonging to the ODO clause, i.e. 'xx'.

Note:
In previous AVB versions, it was necessary to specify one entry in the ODO parameters for each MU/PE field. Adabas Bridge for VSAM version 5.1 requires only one entry (namely the first MU/PE field name) in the ODO parameters for the entire VSAM repeating field.

When multiple ODO clauses are used in one VSAM record, you can use one GEN entry to define all of them:

- Enclose the values in parentheses.
- Use a comma to separate the values for each ODO clause.
- Ensure that the order in which you specify the entries in the ODNAME, ODINDX, and ODMAX parameters is always the same; that is, the first ODNAME entry must belong to the same ODO clause as the first ODINDX and ODMAX entries, and so on.

The following GEN parameters of previous AVB versions are obsolete and are refused by Adabas Bridge for VSAM version 5.1:

ODFMT; ODLEN; ODOFF; ODPOS; OSOFF; OSSIZ

Example 1 : One MU/PE Field

The repeating field PAYMENTS (P1 in Adabas) has a variable number of occurrences:

```
...
05 PMTS          PIC 9(1).
05 PAYMENTS      OCCURS 1 TO 3 TIMES DEPENDING ON PMTS.
10 DATE          PIC 9(6).
10 AMOUNT        PIC 9(6).
10 MEMO          PIC X(40).
...
```

Adabas Field Definition Table:

COBOL Field Name	Level	Name	Length	Format	Options
05 PMTS	02	O1	1	U	NU
05 PAYMENTS OCCURS ...	01	P1			PE(3)
10 DATE	02	AM	6	U	NU
10 AMOUNT	02	AN	6	U	NU
10 MEMO	02	AP	40	A	NU

GEN Entry:

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                  X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,            X
               ODNAME=P1,ODINDX=O1,ODMAX=3,                    X
               FORMAT='KK,AH,GF,O1,P1001—N'
```

Example 2 : Multiple MU/PE Fields

In the INVOICES file, the VSAM repeating field ITEMS has more than 64 occurrences:

```
...
05 ITEM–CNT            PIC 9(3).
05 ITEMS OCCURS 1 TO 150 TIMES DEPENDING ON ITEM–CNT.
10 ITEM–NO    PIC 9(5).
10 ITEM–QTY   PIC 9(3).
10 ITEM–PRC   PIC 9(6).
...
```

Adabas Field Definition Table:

Three PE groups are needed to cover all 150 occurrences.

COBOL Field Name	Level	Name	Length	Format	Options
05 ITEM–CNT	02	O1	1	U	NU
05 ITEMS	01	P1			PE(64)
10 ITEM–NO	02	AB	5	U	NU
10 ITEM–QTY	02	AC	3	U	NU
10 ITEM–PRC	02	AD	6	U	NU
05 ITEMS	01	P2			PE(64)
10 ITEM–NO	02	AE	5	U	NU
10 ITEM–QTY	02	AF	3	U	NU
10 ITEM–PRC	02	AG	6	U	NU
05 ITEMS	01	P3			PE(22)
10 ITEM–NO	02	AH	5	U	NU
10 ITEM–QTY	02	AJ	3	U	NU
10 ITEM–PRC	02	AK	6	U	NU

GEN Entry:

```
MCTAB TYPE=GEN,FN=INVOICES,FNR=15,RECSIZ=2161,                    X
               KEY1=SK,KEYLEN=8,KEYOFF=0,INDXTYP=P,               X
               ODNAME=P1,ODINDX=O1,ODMAX=150,                     X
               FORMAT='SK,A1,AA,A2,O1,P1001–N,P2001–N,P3001–N'
```

Although the COBOL file definition and the Adabas FDT define the fields in the same order, FORMAT references all the level 1 and level 2 fields in the Adabas FDT. This is necessary because PEs must be defined as level 1 and referenced explicitly.

Note that the RECSIZ parameter specifies the longest record possible, which is a record with all 150 occurrences of ITEMS (P1, P2, and P3).

Example 3 : Multiple ODO Clauses in One VSAM Record

In the following example, all kind of arrays are available in one VSAM record. The repeating field FIX–ARR has a fixed number of occurrences, VAR–ARR has a variable number of occurrences, and LONG–ARR has a variable number of occurrences and more than 64 occurrences. The index fields of the ODO clauses must be in the fixed part of the record.

```
...
05 FIX–ARR              PIC X(20) OCCURS 5 TIMES.
05 VAR–CNT              PIC 9(1).
05 LONG–CNT             PIC 9(3).
05 VAR–ARR              OCCURS 1 TO 3 TIMES DEPENDING ON VAR–CNT.
10 VAR1                 PIC 9(5).
10 VAR2                 PIC 9(8).
10 VAR3                 PIC X(20).
05 LONG–ARR             OCCURS 1 TO 160 TIMES DEPENDING ON LONG–CNT.
10 LONG1                PIC 9(6).
10 LONG2                PIC 9(4).
10 LONG3                PIC X(12).
...
```

Adabas Field Definition Table:

COBOL Field Name	Level	Name	Length	Format	Options
05 FIX–ARR OCCURS ...	01	MA	20	A	MU(5)
05 VAR–CNT	01	O1	1	U	NU
05 LONG–CNT	01	O2	3	U	NU
05 VAR–ARR OCCURS ...	01	P1			PE(3)
10 VAR1	02	AA	5	U	NU
10 VAR2	02	AB	8	U	NU
10 VAR3	02	AC	20	A	NU
05 LONG–ARR OCCURS ...	01	P2			PE(64)
10 LONG1	02	AD	6	U	NU
10 LONG2	02	AE	4	U	NU
10 LONG3	02	AF	12	A	NU
05 LONG–ARR OCCURS ...	01	P3			PE(64)

COBOL Field Name	Level	Name	Length	Format	Options
10 LONG1	02	AG	6	U	NU
10 LONG2	02	AH	4	U	NU
10 LONG3	02	AI	12	A	NU
05 LONG–ARR OCCURS ...	01	P4			PE(32)
10 LONG1	02	AJ	6	U	NU
10 LONG2	02	AK	4	U	NU
10 LONG3	02	AL	12	A	NU

GEN Entry:

Because the FIX–ARR is a fixed-length table, no OD* entry is required for this field. The first entry in the OD* parameters belongs to the VAR–ARR and the second to the LONG–ARR. Only the first PE (‘P2’) of the LONG–ARR must be specified with the `ODNAME` parameter.

```
MCTAB TYPE=GEN,FN=EXAMPLE,...,                                          X
      ODNAME=(P1,P2),ODINDX=(O1,O2),ODMAX=(3,160),                      X
      FORMAT='KK,..,MA001–005,O1,O2,P1001–N,P2001–N,P3001–N,P4001–N'
```

Using Adabas Long-Alpha Fields

An Adabas long-alpha field is defined with the LA option in the Adabas ADACMP cards. For more information about the rules that apply when defining a long-alpha field, see the *Adabas Utilities Manual*.

For use with Adabas Bridge for VSAM, a long-alpha field must

- have a fixed length.
- have its field name specified in the format buffer without the length.
- be used in only one format buffer of a multiple record type file.

  If you need long-alpha fields in more than one format buffer of a multiple record type file, you must use another Adabas long-alpha field.
- be positioned before, between, or after the ODO repeating field(s).

The long-alpha field must **not**

- provide a value in the two-byte length field.

  The length field is handled transparently by AVB.
- be a member of an Adabas PE group.
- be an Adabas MU field.
- be nor contain the primary or secondary VSAM keys (Adabas DEs).
- precede nor contain a record type field of a multiple record type file.
- precede nor contain an OCCURS-DEPENDING-ON (ODO) index field.

GEN Parameters for Long-Alpha Fields

Specify the following GEN parameters in the transparency table for long-alpha fields:

Parameter	Specify ...	Value Format	Default
LANAME	the two-character Adabas name(s) of the long-alpha field(s).	xx | (xx,xx,xx,...)	none
LALEN	the fixed length (in bytes) of the long-alpha field(s).	n | (n,n,n,...)	none

When multiple long-alpha fields are used for one VSAM record, use one GEN entry to define all of them:

- Enclose the values in parentheses.
- Use a comma to separate the values for each long-alpha field.
- Ensure that the order in which you specify the entries in the LANAME and LALEN parameters is always the same; that is, the first LANAME entry must belong to the same long alpha field as the first LALEN, and so on.

Example:

In the following example, the three COBOL fields TEXT–1, TEXT–2 and TEXT–3 are migrated into Adabas long-alpha fields.

```
...
05 TEXT–1              PIC X(300).
05 TEXT–2              PIC X(800).
05 TEXT–3              PIC X(500).
...
```

Adabas Field Definition Table:

COBOL Field Name	Level	Name	Length	Format	Options
05 TEXT–1	01	L1	0	A	NU,LA
05 TEXT–2	01	L2	0	A	NU,LA
05 TEXT–3	01	L3	0	A	NU,LA

GEN Entry:

The first entry in the LA* parameters belongs to TEXT–1, the second to TEXT–2, and the third to TEXT–3.

```
MCTAB TYPE=GEN,FN=EXAMPLE,...,                                  X
               LANAME=(L1,L2,L3),LALEN=(300,800,500),           X
               FORMAT='KK,..,L1,L2,L3,...'
```

Defining Multiple Record Types

To define a file with multiple record types to AVB, you must specify in the GEN entry

- the number of record types in the file;
- the record-type field; and
- the Adabas format buffer for each record type.

Multiple Record Type Parameters

Use the following parameters to define a multiple-record-type file:

Parameter	Specify . . .	Possible Values	Default
RECTYPE	whether the file has more than one record type.	N | Y	N
RECTCNT	the number of record types in the file.	number	0

Example:

The STUDENTS file has two record types:

COBOL Field Name	Level	Name	Length	Format	Options
01 STUDENTS	01	GA			
05 KEY	02	KK			
10 ID–NO	03	K1	6	A	UQ
10 RECTYPE	03	II	1	A	
[FIRST RECORD TYPE]	01	GB			
05 FULL–NAME	02	GC			
10 FNAME	03	AA	15	A	NU
10 LNAME	03	AB	20	A	NU
05 ADDRESS	02	GD			
10 STREET	03	AC	25	A	NU
10 CITY	03	AD	20	A	NU
10 STATE	03	AE	2	A	NU
10 ZIP	03	AF	9	U	DE
05 DEPT	02	AG	4	A	
05 ADVISORS	01	MA	20	A	MU(2)
[SECOND RECORD TYPE]	01	GE			
05 TERM	02	AH	6	A	NU
05 CHARGES	02	GF			
10 TUITION	03	AJ	6	U	NU

COBOL Field Name	Level	Name	Length	Format	Options
10 ROOM	03	AK	5	U	NU
10 MEALS	03	AL	5	U	NU
05 PMTS	02	01	1	U	NU
05 PAYMENTS	01	P1			PE
10 DATE	02	AM	6	U	
10 AMOUNT	02	AN	6	U	
10 MEMO	02	AP	40	A	

GEN Entry:

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                          X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,                    X
               ODNAME=P1,ODINDX=01,ODMAX=3,                            X
               RECTYPE=Y,RECTCNT=2,...
```

This example is continued in the next two sections to illustrate the parameters for the record-type field and the format buffers.

Defining the Record-Type Field

Because some fields are common to all record types and others are used only in a specific type, AVB uses the **record-type field** to determine which format to use for a specific record. The following parameters define the record-type field to AVB:

Parameter	Specify . . .	Possible Values	Default
RECIDFB	the name of the Adabas record-type field.	two-character Adabas name	none
RECTBYT	the length (in bytes) of the record-type field in Adabas.	number	1

Parameter	Specify . . .	Possible Values	Default
RECTOFF	the offset (in bytes) from the beginning of the record to the record-type field.	0–32767	0
RECTFMT	whether the record-type indicator (variable *i*) specified in the `RECFMTS` parameter is alphanumeric (A) or numeric (N). This parameter is required when the format ID is numeric.	A | N	A

Example:

In this example, the RECTFMT parameter does not need to be coded explicitly, because the II field is alphanumeric, which is the AVB default.

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                          X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,                    X
               ODNAME=P1,ODINDX=01,ODMAX=3,                            X
               RECTYPE=Y,RECTCNT=2,                                    X
               RECIDFB=II,RECTBYT=1,RECTOFF=6,...
```

Defining the Adabas Format Buffer for Each Record Type

One or more of the following parameters may be used to specify the format buffer for each record type:

Parameter	Specify . . .	Possible Values	Default
RECFMTS	the format buffer for **each** record type up to 255 bytes as required.	see text	none
RECFMTX	the format buffer for **each** record type beyond 255 bytes up to 510 bytes as required.		
RECFMTY	the format buffer for **each** record type beyond 510 bytes up to 765 bytes as required.		
RECFMTZ	the format buffer for **each** record type beyond 765 bytes up to 1020 bytes as required.		

The length of the RECFMTS parameter value cannot exceed 255 bytes. If more bytes are needed to specify the format buffers for all record types, split the specification into as many as four parts. In addition to RECFMTS, use the RECFMTX, RECFMTY, and RECFMTZ parameters as necessary. Code the first segment (up to 255 bytes) as the RECFMTS value, the next segment as the RECFMTX value, and so on.

The values for these parameters are specified using the following format:

RECFMTS=('ix','ix',..)
RECFMTX=('ix','ix',..)
RECFMTY=('ix','ix',..)
RECFMTZ=('ix','ix',..)

—where "i" represents the record type (the value in the record-type field), and "x" represents the Adabas format buffer used to read or update the record.

See the section **Specifying the Adabas Format Buffer** on page 109 for information about specifying the format buffer. For more information about the Adabas format buffer, consult the *Adabas Command Reference Manual*.

Do not put a comma between the record-type and format-buffer specifications. Enclose each specification in apostrophes; use commas without spaces to separate the specifications for different record types. Enclose all the specifications together in parentheses.

If you have described the format buffer for at least one specific value of "i", you can describe the format buffer for other, unspecified record types using the following symbols as "catchall" values for the "i" variable (the record type).

Symbol	Type	Records where the record-type value is . . .
:	high	**greater** than the largest specified value of "i"
@	low	**less** than the smallest specified value of "i"
#	default	otherwise unspecified

Note the following rules when using the symbols for default, low, and high values:

- Code one symbol for **each** byte in the record-type field. For example, if `RECTBYT=4`, code "####" for "i" to express the default value.
- Do not use hexadecimal representation of these symbols in the macro.
- The symbols for high, low, or default values can be used **only** if one or more specific values for the record type are also specified.

- The format-buffer specifications for high-value (i=:) and low-value (i=@) record types must follow those for specific record types.
- The default (i=#) format-buffer specification must be the last one.

Example 1 : Format Buffer Specifications Requiring Less Than 255 Bytes

The last line specifies the format buffers for record types 1 and 2:

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                              X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,                        X
               ODNAME=P1,ODINDX=O1,ODMAX=3,                                X
               RECTYPE=Y,RECTCNT=2,                                        X
               RECIDFB=II,RECTBYT=1,RECTOFF=6,                             X
               RECFMTS=('1KK,GC,GD,AG,MA001—002','2KK,AH,GF,O1,P1001—N')
```

Example 2 : Format Buffer Specifications Requiring More Than 255 But Less Than 510 Bytes

```
MCTAB TYPE=GEN,FN=ACCOUNTS,FNR=18,RECSIZ=2000,                             X
.
.
               RECFMTS=('1AA,AB,AC,AD,AE,AF,AG,AH,AI,AJ,AK,AL,AM,AN,       X
                        AO,AP,AQ,AR,AS,AT,AU,AV,AW,AX,AY,AZ,               X
                        A1,B1,C1,D1,E1,F1,G1,H1,I1,J1,K1,L1,M1,N1',        X
                        '2AA,AB,AC,AD,AE,AF,AG,AH,AI,AJ,AK,AL,AM,AN,      X
                        AO,AP,AQ,AR,AS,AT,AU,AV,AW,AX,AY,AZ,               X
                        A2,B2,C2,D2,E2,F2,G2,H2,I2,J2,K2,L2,M2,N2')
               RECFMTX=('3AA,AB,AC,AD,AE,AF,AG,AH,AI,AJ,AK,AL,AM,AN,       X
                        AO,AP,AQ,AR,AS,AT,AU,AV,AW,AX,AY,AZ,               X
                        A3,B3,C3,D3,E3,F3,G3,H3,I3,J3,K3,L3,M3,N3',        X
                        '4AA,AB,AC,AD,AE,AF,AG,AH,AI,AJ,AK,AL,AM,AN,      X
                        AO,AP,AQ,AR,AS,AT,AU,AV,AW,AX,AY,AZ,               X
                        A4,B4,C4,D4,E4,F4,G4,H4,I4,J4,K4,L4,M4,N4')
```

Example 3 : Default Format Buffer

Specify the format buffers for record types 1 and 2. Specify a default format buffer for all other record types (all “i” values other than 1 or 2).

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                              X
.
.
               RECFMTS=('1KK,GC,GD,AG,MA001—002',                          X
                        '2KK,AH,GF,O1,P1001—N',                            X
                        '#KK')
```

Example 4 : High Value, Low Value, and Default Format Buffers

The record-type field has two bytes. Specify the format buffers for record types 55, 66, and 77. Specify a format buffer for all record types smaller than 55 and another for all record types greater than 77. Specify a default format buffer for all other record types.

```
MCTAB TYPE=GEN,FN=MANYTYPE,FNR=221,RECSIZ=220,                          X
.
.
               RECFMTS=('55A1,A2,HH,JJ','66A1,A2,MM,NN','77A1,A2,TT',  X
                       '@@A1,A2,CC','::A1,A2,ZZ','##A1,A2,GA,P1')
```

Using PREFETCH

During sequential processing of batch and CICS applications, AVB uses the Adabas PREFETCH option to enhance performance. The batch implementation is separate from the batch ADARUN PREFETCH.

For files with one record type, PREFETCH obtains multiple records for each Adabas call. For files with multiple record types, AVB prefetches only the record-type fields.

AVB uses the ISN buffer in AVB-controlled storage areas to implement PREFETCH. By controlling the size of the AVB ISN buffer, AVB controls the number of records or record-type fields prefetched per call.

Depending on the number of records and the size of the record-type field, you may need to specify a larger ISN buffer size in the PREFSZE parameter.

Parameter	Specify . . .	Minimum	Maximum	Default
PREFSZE	the size (in kilobytes) of the ISN buffer.	0	32	0

Use the PREFSZE parameter to specify the size (in kilobytes) of the ISN buffer to be allocated for record prefetching. Except for files with multiple record types, PREFSZE=0 (the default) disables AVB PREFETCH. Batch users can disable PREFETCH and run the batch ADARUN PREFETCH to benefit from the larger ISN buffer specified by ADARUN parameters.

For ESDS and RRDS files, disable PREFETCH (that is, use the default PREFSZE=0).

PREFETCH is **required** for multiple-record-type files. If you specify PREFSZE=0 for a multiple-record-type file, AVB defaults to a buffer size of 1K and prefetches the number of record-type fields that fit in the 1K buffer. ADARUN PREFETCH offers no advantage for these files.

The maximum number of records or record-type fields prefetched is calculated as

(n • 1024) / (r + 16)

—where "n" is the PREFSZE value (in kilobytes) and "r" is the length (in bytes) of the record or record-type field.

This calculation estimates the number of records or record-type fields that will fit in the specified ISN buffer size. If the estimate results in too many records or record-type fields being returned for the buffer size, AVB adjusts the PREFSZE value automatically.

Note:
When PREFETCH is used, the value for the ADARUN LU parameter may need to be increased.

Example:

```
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                          X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,                    X
               ODNAME=P1,ODINDX=O1,ODMAX=3,                            X
               RECTYPE=Y,RECTCNT=2,                                    X
               RECIDFB=II,RECTBYT=1,RECTOFF=6,                         X
               RECFMTS=('1KK,GC,GD,AG,MA001—002',                      X
                        '2KK,AH,GF,O1,P1001—N'),                       X
               PREFSZE=10
```

Incorporating the VSAM Password

If you are using a VSAM password at open time and want to use the same password for the Adabas file, specify the PWD parameter:

Parameter	Specify . . .	Possible Values	Default
PWD	the password to be inserted in Adabas calls for access to the file.	password	none

To incorporate the password in AVB processing

1. Add the `PWD` parameter to the GEN entry for the file, specifying the password contained in the ACB.
2. Update the Adabas database to indicate file security.

Overriding the Global Adabas DBID

Note:
AVB requires that all Adabas files used by a transaction or application program exist in the same physical database.

Parameter	Specify . . .	Minimum	Maximum	Default
DBID	an Adabas DBID for the session.	0	65535	0

For information about setting the global Adabas DBID, see page 100. The maximum DBID is 65535.

If you want to override the global DBID and specify a different DBID for a particular file, specify the DBID parameter in the GEN entry of the file.

AVB selects the Adabas DBID to be inserted into the Adabas control block as follows:

1. If the GEN entry for a file specifies a value in the DBID parameter, that DBID is used.
2. If the GEN entry specifies DBID=0 (the default), the global DBID specified in the INITIAL statement is used.
3. If the INITIAL statement also specifies DBID=0 (the default), the DBID specified in the Adabas link routine is used.

Specifying File Status at Startup

Parameter	Specify . . .	Possible Values	Default
OCIND	whether the file will be open (O) or closed (C) when starting AVB (CICS only). If V is specified, the file is bypassed by AVB processing and opened by native VSAM.	C | O | V	C

You need to code this parameter only if you want a file to be open at AVB startup (OCIND=O). Use OCIND=V when you want a file to be processed by VSAM and not by AVB.

Specifying the AVB Response to the RESET ACB Indicator

Parameter	Specify . . .	Possible Values	Default
RESET	the action AVB is to take for a file-open request when the RESET ACB indicator is set.	EMPTY | APPEND | DELETE	EMPTY

Use the RESET parameter to specify how AVB will handle a request to open a file when the RESET ACB indicator is set (OPEN output in COBOL always sets the RESET indicator):

Value	Specifies that . . .
EMPTY	(the default) the open request is to be rejected if the file is not empty.
APPEND	records are to be added at the end of the existing file.
DELETE	the existing records are to be deleted.

Note:
DELETE may be time-consuming for large files. Software AG recommends that you add Adabas file delete and define steps before application execution rather than allowing AVB to refresh the file.

Example:

```
MCTAB TYPE=GEN,FN=EMPLOYEE,FNR=11,RECSIZ=65,                              X
               KEY1=SK,KEYLEN=9,KEYOFF=0,INDXTYP=P,                       X
               FORMAT='NA,SK,DP,TI,TB',GLBLFID=DYNAMIC,RESET=APPEND
```

Completed Examples

```
MCTAB TYPE=INITIAL,DBID=100,GPWD=AVB

MCTAB TYPE=GEN,FN=EMPLOYEE,FNR=11,RECSIZ=65,                              X
               KEY1=SK,KEYLEN=9,KEYOFF=0,INDXTYP=P                        X
               FORMAT='NA,SK,DP,TI,TB',RESET=APPEND
MCTAB TYPE=GEN,FN=INVOICES,FNR=15,RECSIZ=2161,                            X
               KEY1=SK,KEYLEN=08,KEYOFF=0,INDXTYP=P,                      X
               FORMAT='SK,A1,AA,A2,O1,P1001-N,P2001-N,P3001-N,            X
               ODNAME=P1,ODINDX=O1,ODMAX=150,                             X
               KEYB=RB,DBID=121
MCTAB TYPE=GEN,FN=INVCUST,FNR=15,RECSIZ=2161,                             X
               KEY1=A1,KEYLEN=04,KEYOFF=08,INDXTYP=S,                     X
               FORMAT='SK,A1,AA,A2,O1,P1001-N,P2001-N,P3001-N,            X
               ODNAME=P1,ODINDX=O1,ODMAX=150,                             X
               DBID=121
MCTAB TYPE=GEN,FN=INVNAME,FNR=15,RECSIZ=2161,                             X
               KEY1=A2,KEYLEN=25,KEYOFF=32,INDXTYP=S,UNIQUE=N,            X
               FORMAT='SK,A1,AA,A2,O1,P1001-N,P2001-N,P3001-N,            X
               ODNAME=P1,ODINDX=O1,ODMAX=150,                             X
               DBID=121
MCTAB TYPE=GEN,FN=STUDENTS,FNR=18,RECSIZ=301,                             X
               KEY1=SK,KEYLEN=7,KEYOFF=0,INDXTYP=P,                       X
               ODNAME=P1,ODINDX=O1,ODMAX=3,                               X
               RECTYPE=Y,RECTCNT=2,                                       X
               RECIDFB=II,RECTBYT=1,RECTOFF=6,                            X
               RECFMTS=('1KK,GC,GD,AG,MA001-002',                         X
                        '2KK,AH,GF,O1,P1001-N'),                          X
               PREFSZE=10CTAB TYPE=END

MCTAB TYPE=ETOPNL,UPD='11,15,18'
```

Assembling and Linking the Transparency Table

To assemble and link the transparency table

1. Modify the jobs ASMTABB (for batch processing) and/or ASMTABC (for CICS).
   - Supply job card information and modify other JCL statements as required for your site.
   - In the symbolic parameter M, supply the name of the source module that contains the MCTAB macro statements.

     For batch processing, you can create different source versions of the table to describe different views and/or VSAM files for your applications.
   - In the symbolic parameter L and in the NAME linkage-editor control statement, supply the name of the load module to be created for the transparency table:
     - For CICS processing, the name **must** be AVBTAB.
     - For batch processing, the load module can have any name; this name will be specified in an AVBUSER control statement in the job step that activates AVB.
   - Note the following:
     - Do not use the RENT or REUS linkage-editor parameters.
     - The RMODE for the load module can be 24 or ANY.
     - For CICS, be sure the AVBTAB module is linked into a CICS DFHRPL concatenated load library.
2. Run the jobs ASMTABB and/or ASMTABC.

7

CREATING THE OPTIONS TABLE

The **options table** defines the AVB system dependencies and user options for your site. It is required in all environments where AVB is used.

This chapter provides instructions for users converting from AVB version 3, for users converting from AVB version 4, and for new users. It then describes all required and optional parameters.

It is recommended that you prepare a single options table and link it into all environments where AVB will be used. If you choose to use multiple options tables, you must have multiple load libraries because each copy of the load module for the options table **must** be named AVBOPT.

Each time you change the options table (AVBOPT), you must reassemble and relink it. Refer to the installation chapter for your particular environment for instructions.

- The AVBOPT table **must** be reassembled and relinked using the AVB version 5.1 MVSSRCE library concatenated first in the assembly SYSLIB.
- Do **not** use the AVBOPT load module from a previous version of AVB when running with AVB version 5.1.

Converting from AVB Version 3

To convert your version 3 options table to version 5

Make the following changes to your AVBOPT source table:

- Remove the following parameters, which are no longer used: CWAOFF; CWAYN; HPO; IDC13; REUSIO; SVC; USRABND
- Set CSECT=YES and OPSYS={MVSXA | OS390 | VSEESA}.

  CSECT=YES (the default) is required to generate the options table.

  OPSYS=MVSXA (the default) can be used for OS/390; however, the option OS390 is available. The option VSEESA is available to support VSE/ESA systems.

  Ensure that your AVBOPT source table does not contain conflicting values and that these values are not deleted or changed in the MCOPT macro itself.
- Set VER=5.1.s, where ‘s’ is the system maintenance (SM) level of AVB version 5.1.

Converting from AVB Version 4

To convert your version 4 options table to version 5

Make changes to your AVBOPT source table based on the following parameter changes:

Parameter	Description of Changes
ADABNAM	The default value for the batch Adabas link routine is ADALNK. Enter a value that matches the name of the batch Adabas link routine to be invoked by AVB version 5.1.
ADALNAM	The default value for version 4.1 was AVBADA; for version 4.2 and 5.1, it is ADABAS. Enter a value that matches the name of the CICS command-level link routine to be invoked by AVB version 5.1.
ADATNAM	Specify the name of the Adabas CICS task-related user exit (TRUE) program associated with the Adabas CICS command-level link routine. AVB requires the Adabas CICS TRUE for proper operation. The default value is ADATRUE. Enter the correct value.

Parameter	Description of Changes
ADAVER	The default value is “7”. Enter a value that matches the version of Adabas running with AVB version 5.1.
OPSYS	OPSYS={MVSXA | OS390} may be specified for AVB version 5.1 for z/OS or OS/390 installations, or OPSYS=VSEESA for VSE/ESA installations.

New Users

To create your version 5 options table

Modify the source member AVBOPT.

Software AG recommends that you define all options for your site at this time, including optional parameters.

As a **minimum,** set the following required parameters:

- all environments:
  ADAVER; CSECT=YES; MAXFILE; VER=5.1.s; OPSYS={MVSXA | OS390 | VSEESA}
- CICS environments: ADALNAM; ADATNAM; MAXTRAC

Specify as well either ETRC or INITRC. These parameters are optional, but one or the other is recommended, depending on the type of application logic generally used at your site. See page 142 for more information.

Default values are set in the MCOPT table. Ensure that these parameters are not deleted or changed in the MCOPT macro itself.

AVBOPT Parameters

The default values provided in this section are the values coded in the MCOPT macro distributed by Software AG. If the source code for the macro is modified at your site, these default values could change.

All Environments

ADAVER

Parameter	Specify . . .	Possible Values	Default
ADAVER	the Adabas version number.	6 | 7	7

CSECT

Parameter	Specify . . .	Possible Values	Default
CSECT	whether to generate the options table.	NO | YES	YES

The default YES is required to generate the table.

CUST

Parameter	Specify . . .	Possible Values	Default
CUST	the customer name or ID.	‘name’	none

The value can be up to 20 characters enclosed in single quotes.

DBGNZO

Parameter	Specify . . .	Possible Values	Default
DBGNZO	whether only nonzero Adabas response codes are saved in the debug trace table.	NO | YES	NO

All entries in the debug trace table are saved in wrap-around fashion. The size of the table is based on the value of the MAXTRAC parameter.

DYNMGFID

Parameter	Specify . . .	Possible Values	Default
DYNMGFID	the level of support for the dynamic global format ID.	OFF | ON | ALL	ON

Software AG strongly recommends that you use Adabas global format IDs for online applications that share a common record layout. Global format IDs improve performance because Adabas does not have to rebuild the format buffer for each Adabas call.

Value	Meaning
ON	(the default) Dynamic global format ID support is active, but only for those files defined with the MCTAB parameter GLBLFID=DYNAMIC. See page 111 for more information.
OFF	**No** AVB bridged files have dynamic global format ID support.
ALL	**All** AVB bridged files have dynamic global format ID support.

EOTRC

Parameter	Specify . . .	Possible Values	Default
EOTRC	whether an RC command is issued at end of task, even for read-only processes.	NO | YES	NO

This parameter can benefit an application that browses files for long periods because it frees resources at the end of each task.

MAXFILE

Parameter	Specify . . .	Possible Values	Default
MAXFILE	the maximum number of bridged files that may be opened by a job or transaction.	1–200	150

An AVB file table is allocated with 160 bytes for each entry.

MAXTRAC

Parameter	Specify . . .	Possible Values	Default
MAXTRAC	the number of entries to be allocated for the debug trace table.	1–500	400

The default value provided is adequate for CICS environments. Software AG recommends that batch users specify an appropriate value for their environments.

MOPNA

Parameter	Specify . . .	Possible Values	Default
MOPNA	whether AVB is to allow multiple consecutive OPENs for a file without an intervening CLOSE.	NO | YES	NO

OPSYS

Parameter	Specify . . .	Possible Values	Default
OPSYS	the operating system for system-dependent AVB functions.	MVSXA | OS390 | VSEESA	MVSXA

R145CNT

Parameter	Specify . . .	Possible Values	Default
R145CNT	AVB processing after an Adabas response code 145.	0–999 | RETURN	0

Response code 145 indicates that an update request has been issued for a record held by another user or transaction. AVB processing after response code 145 can affect system performance in a competitive update environment:

Value	Meaning
0	(the default) The application is to wait until either the requested record is made available for update or the Adabas transaction timer expires.
1–999	The number of times AVB will reissue the update command before returning response code 145 to the application (as an appropriate VSAM feedback code).
RETURN	AVB returns control to the application immediately, without waiting or retrying the command. This option gives the best overall system performance in highly competitive update environments.

For more information about Adabas response code 145 and competitive updating, consult the *Adabas Command Reference Manual*.

VARLRECL

Parameter	Specify . . .	Possible Values	Default
VARLRECL	whether VSAM variable records are supported.	OFF | ON | ALL	OFF

When this parameter is set to ON or ALL, AVB defines an additional four-byte binary field at the beginning of every Adabas record in an AVB file that will support VSAM variable records. The first two bytes contain the record length; the next two bytes contain low values.

This setting must be reflected in the transparency table (MCTAB): the Key Offset field for that AVB file will be offset four bytes to include the length of the Length field itself, and the record size will be similarly affected.

Value	Meaning
OFF	(the default) AVB files do not support VSAM variable records.
ON	VSAM variable record support is implemented for certain AVB files only. For these files, the transparency table (MCTAB) parameter VARLRECL must also be set to ON. See page 103 for more information.
ALL	VSAM variable record support is implemented for **all** AVB files.

Batch Environments Only

ADABNAM

Parameter	Specify . . .	Possible Values	Default
ADABNAM	the name of the batch Adabas link routine.	see text	ADALNK

Enter a value that matches the name of the batch Adabas link routine to be invoked by AVB version 5.1.

COBPARM

Parameter	Specify . . .	Possible Values	Default
COBPARM	whether system parameters are passed using the JCL EXEC statement.	NO | YES	NO

Value	Meaning
NO	(the default) Specify this value for all languages other than COBOL.
YES	Specify this value **only** when using COBOL. Save-area pointers passed to a COBOL application program are then modified so that COBOL run-time modules can extract extended system parameters from the JCL EXEC statement.

CICS Environments Only

ADALNAM

Parameter	Specify . . .	Possible Values	Default
ADALNAM	the entry-point name of the Adabas CICS command-level link routine used by AVB.	entry-point-name	ADABAS

The installation chapter describes the installation of the command-level link routine.

ADATNAM

Parameter	Specify . . .	Possible Values	Default
ADATNAM	the name of the Adabas CICS task-related user exit (TRUE) program.	module-name	ADATRUE

This program is required with the Adabas CICS command-level link routine.

ASYNCL

Parameter	Specify . . .	Possible Values	Default
ASYNCL	whether AVBCICS3 issues a CL command when invoked by asynchronous CICS transactions.	NO | YES	NO

Note:
The parameter has no effect on tasks associated with terminals.

AVBCICS3 is the CICS TRUE for the CICS syncpoint program (SPP).

Value	Meaning
NO	(the default) AVBCICS3 issues an ET command when invoked by asynchronous CICS transactions. Specify this value if CICS asynchronous tasks generally start infrequently and run a long time, such as a task that starts when CICS comes up and remains active with the same task ID during the entire CICS session.
YES	AVBCICS3 issues a CL command when invoked by asynchronous CICS transactions. Specify this value if CICS asynchronous tasks generally start and finish repeatedly during the session, acquiring different task IDs each time. The CL command frees Adabas resources, such as command and user queue elements, that would otherwise remain active and deplete Adabas work pool space during the CICS session. However, the CL command produces more overhead than the ET command and must therefore be used judiciously.

CTRDMP

Parameter	Specify . . .	Possible Values	Default
CTRDMP	whether to generate snapshot dumps for AVB control blocks and work areas during the execution of CICS transactions using AVB.	NO | YES	NO

This value can be changed using the AVZP transaction.

ETRC and INITRC

Parameter	Specify . . .	Possible Values	Default
	whether AVB generates an Adabas RC command		
ETRC	—with each ET or BT command.	NO | YES	NO
INITRC	—at the start of each transaction; that is, before any other Adabas command is issued.	NO | YES	NO

Although Adabas RC commands help prevent resource conflicts, they carry overhead. In an effort to minimize the number of RCs issued, Software AG recommends that you specify a value for these parameters depending on the type of application logic used most at your site:

Parm	Value	CICS transactions executed with AVB . . .
ETRC		generally **issue command sequences that are** logically
	NO	—related, such as BROWSE with subsequent update. Internal resources remain intact after the completion of the ET command, reducing Adabas overhead.
	YES	—independent of the previous command sequence. The RC command releases internal Adabas resources, which helps to prevent resource conflicts.
INITRC		**are themselves** logically
	NO	—independent. Fewer Adabas commands are issued, so performance is improved.
	YES	—related and need resources that were used by a transaction just completed. The RC command releases internal Adabas resources.

See the *Adabas Command Reference Manual* for more information about the RC command.

LOGO

Parameter	Specify . . .	Possible Values	Default
LOGO	whether AVB displays the logo screen with the AVB1, AVTR, and AVZP transactions.	NO | YES	YES

MAXCTSK

Parameter	Specify . . .	Possible Values	Default
MAXCTSK	the maximum number of concurrent CICS tasks	see text	500

The MAXCTSK parameter defines a table that comprises a 20-byte header and 8-byte entries. The table size is therefore

20 + (value of (MAXCTSK) * 8)

The table is allocated by AVBCICS0 during AVB version 5.1 initialization. Entries are obtained during file control BLUE processing by AVBCICS2. Each entry used is freed at end-of-task by AVBCICS3. The table is released from CICS storage when AVB version 5.1 is deactivated by AVBCICS9.

SUPRPWD

Parameter	Specify . . .	Possible Values	Default
SUPRPWD	whether to suppress the password requirement for the AVB1 transaction.	NO | YES	NO

8

CICS OPERATION

After you have migrated your VSAM application to Adabas, you can invoke AVB and run your VSAM application programs against Adabas databases.

This chapter discusses the CICS operation of AVB. It provides a detailed discussion of AVB logic: how AVB is activated and deactivated; and how it processes requests.

Activating / Deactivating AVB

Under CICS/TS, the AVBCICS0 program establishes the CICS global and task-related user exits, which process VSAM file requests and convert the requests for files with DD names (DTF names in VSE/ESA) in the AVB transparency table to Adabas calls.

AVBCICS0 is executed during PLTPI processing, by the AVB1 transaction, or by AVB Online Services:

- If you created an entry for the AVBCICS0 module in the CICS start-up table (PLTPI), AVB starts automatically when CICS is started. This is the recommended procedure.
- If your site does not allow this, start AVB with the AVB1 transaction or with AVB Online Services. AVB1 is described on page 146; AVB Online Services is described in the chapter **AVB Online Services** starting on page 163.

AVBCICS0 also activates the CICS file control global user exits (SPI and file control). These exits are given control by CICS during the processing of file requests like START BROWSE and during OPEN and CLOSE processing by applications or by the CEMT transaction.

The AVBCICS9 module closes files that were opened by AVB and deactivates the CICS file control global user exits. AVBCICS9 is executed during PLTSD processing, by the AVB1 transaction, or by AVB Online Services.

See page 31 (page 66 for VSE/ESA) for information about adding the PLTPI and PLTSD entries.

In addition, AVB version 5.1 provides the AVBCICSZ module, which is installed as a CICS “shutdown” task-related user exit (TRUE). AVBCICSZ receives control when CICS is terminated by CEMT P,SHUT or CEMT P,SHUT,I commands from the operator console or from a CICS operator:

- If the shutdown is orderly, AVBCICSZ passes control to AVBCICS9 to deactivate AVB.
- If the shutdown is catastrophic, AVBCICSZ writes a message to the console indicating that AVB could not be terminated normally. If the normal CICS syncpoint processing is not possible in such cases and the CICS is not restarted before the Adabas transaction timers expire, Adabas transactions are backed out by the Adabas nucleus.

Executing the AVB1 Transaction

To manually activate or deactivate AVB

1. Execute the AVB1 transaction.

   Note:
   Each time the AVB1 transaction is issued, you may be required to enter a password. The password is defined in the MCTAB GPWD parameter described on page 101.

   The AVB1 screen appears:

```
AVB V5.1.1 – OS/390 2.08 – CICS/TS 5.3
AVBridge V5.1.1 12/08/00                          12/15/00  17:48:58     AVB1A

Status:   AVB1000 – AVBridge Activated

Function:           ON/OFF/(STATS|FILES)/END
Options :           OPEN/CLOSE/VSAM/filename
Password:
```

   The first line shows the AVB version; the operating system, version, and release level; and the CICS version and release level. The current date and time appear on the next line.

   The status line shows whether AVB is active.

2. Type ON (activate) or OFF (deactivate) as required on the Function line and press ENTER.
3. Exit the AVB1 transaction by typing END on the Function line and pressing ENTER.

CICS Request Processing

OPEN Requests

AVB processes OPEN requests by a CICS application in the following steps:

1. A CICS application, utility, or service issues an OPEN request for a file. The CICS file control domain receives control.
2. CICS file control passes the request to AVBCICSA through the system programming interface's (SPI) global user exit (GLUE).
3. AVBCICSA determines whether the request is for a file bridged by AVB.
   - If the request is not for a bridged file, AVBCICSA returns control immediately to CICS file control.
   - If the request is for a bridged file, AVBCICSA acquires and initializes storage areas needed for AVB processing.
4. AVBCICSA then requests that the AVB Adabas interface routine AVBPROG issue an Adabas LF command to produce the Adabas FDT information necessary to handle any repeating and "occurs depending on" data that may be part of the application's view of the original VSAM record.
5. When the LF command is successful and the necessary FDT data is available, AVBCICSA
   - bypasses CICS file control; and
   - sets the file status in the AVB transparency table to "open enabled".
6. CICS drives the SPI GLUE post-exit point. AVBCICSA interprets any VSAM feedback codes or Adabas response codes and sets the CICS EIBRCODE, EIBRESP, and EIBRESP2 fields accordingly to provide the necessary feedback to the CICS application.

The file OPEN process under CICS is illustrated in the following figure:

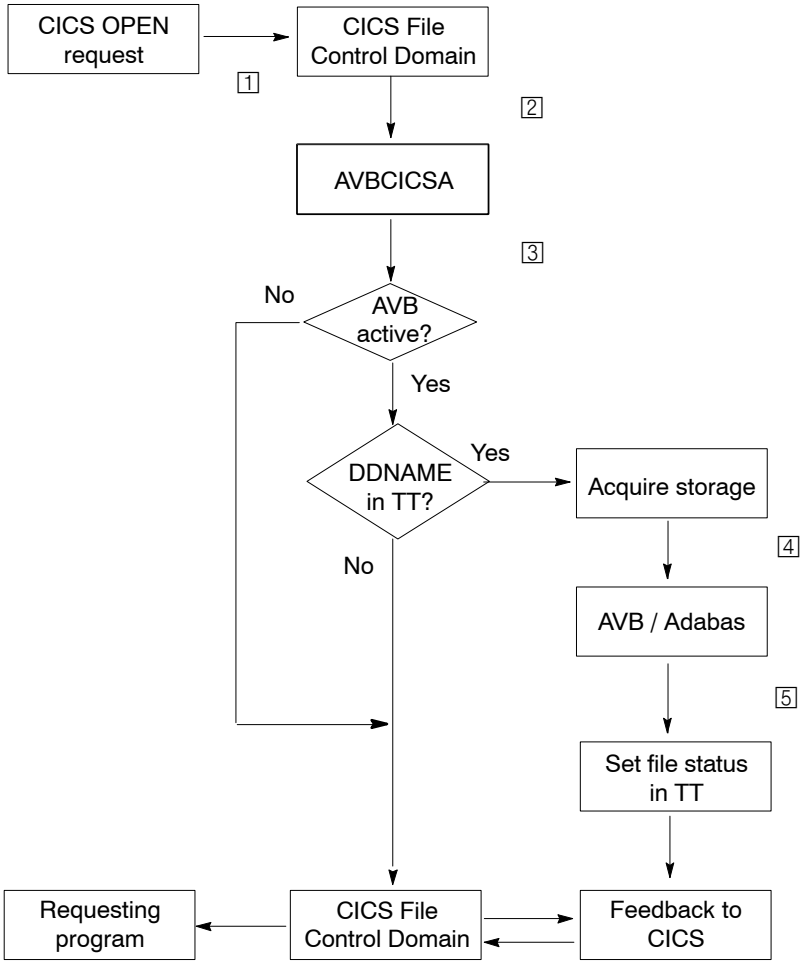


Figure 8-1: File OPEN Processing under CICS

File Requests

1. An application program issues a VSAM file request (STARTBROWSE, READNEXT, etc.). The CICS file control domain receives control.
2. CICS file control passes the request to AVBCICS2.
3. AVBCICS2 determines whether the request is for a file bridged by AVB.
   - If the request is not for a bridged file, AVBCICS2 returns control immediately to CICS file control.
   - If the request is for a bridged file, AVBCICS2 acquires and initializes storage areas needed for AVB processing (first request for a CICS task) or reuses storage already acquired (subsequent requests for the same CICS task).
4. If AVB has opened the file, AVBCICS2 invokes the AVB request handling TRUE, AVBREQT. AVBREQT then invokes AVBPROG, which translates the request into an Adabas call and invokes the Adabas CICS command-level link routine.
5. When AVBCICS2 handles the file request instead of VSAM, control is passed directly to the post-exit processing in AVBCICS2 through a bypass option so that VSAM is **not** invoked for the request.
6. When the request has been fulfilled by VSAM or Adabas, CICS file control regains control.
7. CICS file control passes control to AVBCICS2, which determines whether the request results are from a bridged file.
   - If the results are not from a bridged file, AVBCICS2 returns control immediately to CICS file control.
   - If the results are from a bridged file, AVBCICS2 modifies response codes in the CICS EIB as necessary. AVBCICS2 then returns control to CICS file control.
8. CICS file control sets all necessary control blocks and data areas (including the EIB and the I/O areas that contain the returned records) and returns control to the requesting program.

The AVB file access operation under CICS is illustrated in the following figure:

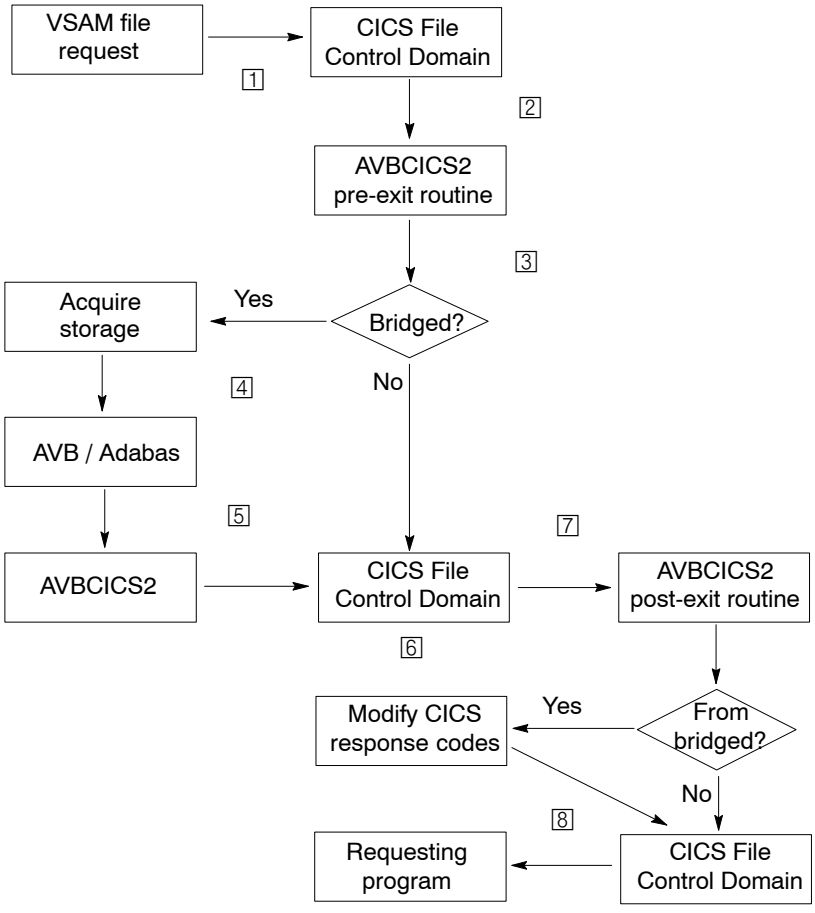


Figure 8-2: File Request Processing under CICS

Implicit OPEN Requests

Under CICS/Transaction Server (TS), AVB intercepts both implicit and explicit OPEN requests. The intercepts permit a mixture of VSAM processing and AVB processing for files in a running CICS system.

For an implicit OPEN where a VSAM file request is issued for a file that has not been opened, the file-state process is nested within the file-request process. The steps are summarized below:

1. An application program issues a VSAM file request (STARTBROWSE, INSERT, etc.). The CICS file control domain receives control.
2. CICS file control passes the request to AVBCICSA.
3. AVBCICSA determines whether the request is for a file bridged by AVB.
   - If the request is not for a bridged file, AVBCICSA returns control immediately to CICS file control.
   - If the request is for a bridged file, AVBCICSA acquires storage areas for AVB **file-request** processing. AVBCICSA bypasses CICS file control at the SPI exit point.
4. CICS file control then drives the file control GLUE where AVBCICS2 evaluates the request.
   - If the results are not from a bridged file, AVBCICS2 returns control immediately to CICS file control.
   - If the request is for a bridged file, AVBCICS2 interprets the request and invokes AVBREQT, which calls AVBPROG to “translate” the request to an Adabas call.
5. AVBPROG invokes the Adabas CICS command-level link routine, which in turn passes the request and data to Adabas.
6. Control returns to AVBPROG usually after a CICS WAIT; AVBPROG then returns control to AVBCICS2 passing data and response information.
7. AVBCICS2 then bypasses CICS file control and is invoked again at the file control post-exit point.
8. AVBCICS2 sets appropriate values in the CICS EIB, sets any necessary I/O pointers, etc.
9. When the request has been fulfilled by VSAM or Adabas, CICS file control regains control.

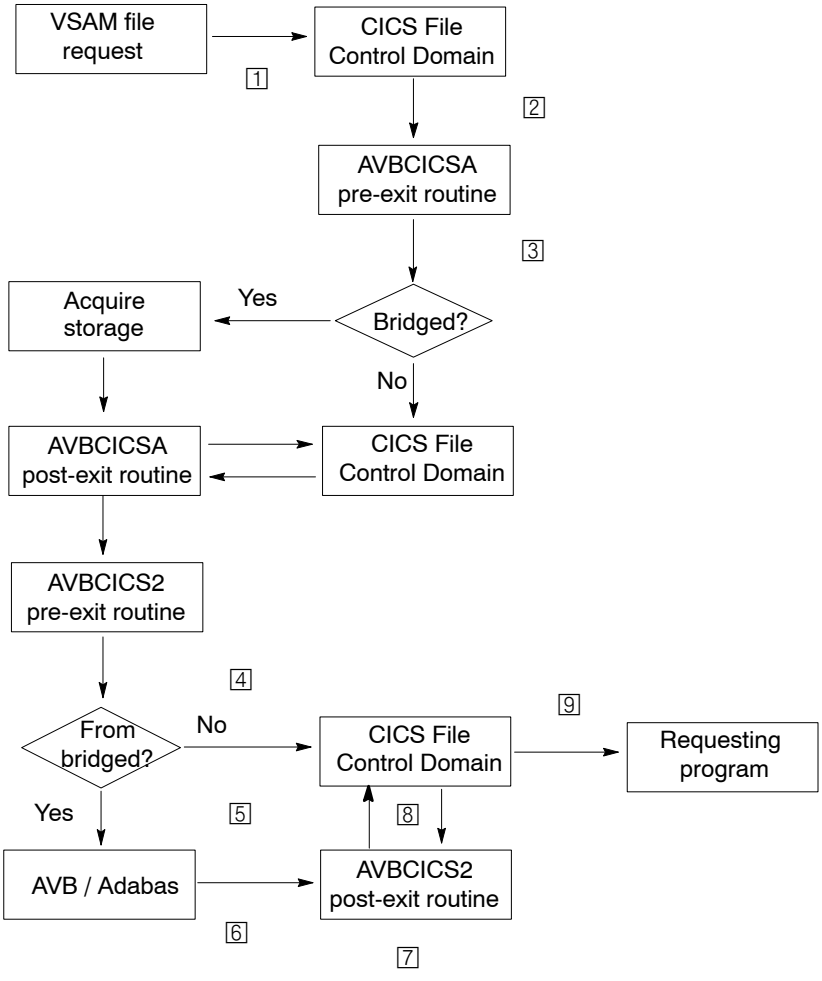


Figure 8-3: Implicit OPEN Request Processing under CICS

CICS SYNCPOINT and SYNCPOINT ROLLBACK Requests

AVB converts CICS SYNCPOINT and SYNCPOINT ROLLBACK requests to Adabas ET and BT commands, respectively.

A CICS application can generate Adabas ET/BT commands by issuing the EXEC CICS SYNCPOINT or EXEC CICS SYNCPOINT ROLLBACK commands.

AVB generates an Adabas ET command at normal end-of-task because CICS issues a SYNCPOINT request at CICS task termination. When a transaction ends abnormally and CICS automatically issues a SYNCPOINT ROLLBACK request, AVB generates an Adabas BT command.

CICS VSAM File Requests

Under CICS, AVB processes a VSAM file request against the corresponding Adabas file when all the following conditions are satisfied:

- The file is defined in the Adabas transparency table;
- The OCIND (field fileoc) in the AVB transparency table is set to either O or C;
- The VSAM file has a valid entry in the FCT or CSD;
- AVB is activated.

Note:
You must activate AVB before you open the file. The VSAM file must not be open when you activate AVB.

Coding CSD Entries for Bridged Files

OS/390 or z/OS

A CSD or FCT entry is required for each file to be processed through AVB:

- Specify STATUS(UNENABLED) and OPENTIME(FIRSTREF) to ensure that the VSAM file is not opened before AVB is activated.
- Specify LSRPOOLID(NONE).

AVB honors all the processing values set in the FCT except EXCLUSIVE (for example, READ VALID, BROWSE VALID are honored).

The following entry defines a test file used to verify the AVB installation:

```
DEFINE FILE(SYS021) GROUP(AVBFILES)
DESCRIPTION(AVB COBOL PRIMARY KSDS IVPFILE)
       LSRPOOLID(NONE)
       DSNSHARING(ALLREQS) STRINGS(4) RECORDSIZE(80) KEYLENGTH(7)
       STATUS(UNENABLED) OPENTIME(FIRSTREF) DISPOSITION(SHARE)
       DATABUFFERS(5) INDEXBUFFERS(4) TABLE(NO) RECORDFORMAT(V)
       ADD(YES) BROWSE(YES) DELETE(YES) READ(YES) UPDATE(YES)
       JOURNAL(NO) JNLREAD(NONE) JNLSYNCREAD(NO) JNLUPDATE(NO)
       JNLADD(NONE) JNLSYNCWRITE(NO) RECOVERY(NONE) FWDRECOVLOG(NO)
       BACKUPTYPE(STATIC)
```

VSE/ESA

A CSD entry is required for each file using AVB, even if the original corresponding VSAM file no longer exists.

AVB honors all the processing values set in the CSD except EXCLUSIVE (for example, READ VALID and BROWSE VALID are honored).

The following items apply to each file to be processed through AVB under CICS:

- It is recommended that the CSD entry specify UNENABLED,CLOSED to ensure that the VSAM file is not opened before AVB is activated.
- Specify LSRPOOL=NONE.

INQUIRE FILE and SET FILE Requests

AVB uses the CICS SPI exit points under CICS/TS 1.1 or above. This allows AVB to intercept system programming INQUIRE FILE or SET FILE requests from application programs or CICS utilities such as CEMT.

As a consequence, it is no longer necessary to keep dummy VSAM file definitions as part of the CICS execution JCL. If AVB is active, it intercepts the INQUIRE FILE or SET FILE requests and bypasses CICS file control and CICS catalog processing.

Note:
If you intend to switch between bridged and nonbridged access to files in a CICS, then the VSAM file definitions are required in your CICS JCL when you open and access files with native VSAM requests.

Opening and Closing Files

To process a VSAM file request against the corresponding Adabas file, AVB **must** be active.

If a VSAM file is opened before AVB is active, and you want a transaction to use the Adabas file, first set the VSAM file to CLOSED,UNENABLED using the CICS transaction CEMT SET FI(filename); then start AVB.

A file is opened automatically when AVB starts if the OCIND parameter in the transparency table is set to “O” for the file. For information about the OCIND parameter, see page 129.

To open or close files manually while AVB is active, use the AVB1 transaction. See the section **Displaying and Updating File Information** on page 159 for more information.

You can also use AVB Online Services to open and close the files. For more information, see the chapter **AVB Online Services** starting on page 163.

When a file is opened, either automatically or manually, AVB sets the FCT open flag, which is required for command-level processing.

If a file is closed (status “C”), AVB rejects VSAM requests and does not forward them to VSAM. To process a request against the VSAM file instead of the Adabas file, use the AVB1 transaction to set the file status to “V” (VSAM); then use the CEMT SET transaction to set the VSAM file to OPEN.

Note:
To subsequently process a request against the Adabas file instead of the VSAM file, first use the CEMT SET transaction to set the VSAM file to CLOSED,ENABLED and then use the AVB1 transaction or AVB Online Services to open the Adabas file.

If the module AVBCICS9 has been defined in the CICS PLTSD table, files used by AVB are closed before AVB shuts down.

Using AVB5

Files may also be opened/closed using the AVB5 transaction. The AVB5 transaction differs from the AVB1 transaction in the following ways:

- Its only function is to open and close files.
- It does not prompt the user for input, but instead accepts parameters from the terminal or console when the transaction is invoked.

- It allows you to open or close a single file, all files, or a group of files as defined in the AVBCICSG group table.

Examples:

Opening and closing single files:

```
AVB5 OPEN,FILE=filename
AVB5 CLOSE,FILE=filename
```

Opening and closing all files:

```
AVB5 OPEN,ALL
AVB5 CLOSE,ALL
```

Opening and closing files in a group:

```
AVB5 OPEN,GROUP=groupname
AVB5 CLOSE,GROUP=groupname
```

Notes:

1. *To use the GROUP parameter, the group table AVBCICSG must first be assembled, linked, and made available to CICS. See the section* ***Creating the Group Table*** *on page 157 for more information.*
2. *The AVB5 transaction is not password-protected. Therefore, you may want to secure AVB5 at the transaction level.*

AVB5 Status Message

On completion, the AVB5 transaction returns a one-line status message to the issuing terminal or console:

- If the request was successful, the message “OK” is returned;
- If the request encountered any errors, the message “ERROR” is returned with the error specified in parentheses.

Creating the Group Table for AVB5

The group table assigns files to groups. This allows the AVB5 CICS transaction to open or close files by group name. The group table is only used by AVBCICSG and is not necessary for the Adabas Bridge for VSAM to function.

Each group has a name (up to 8 bytes) and can have up to 40 filenames associated with it. Any number of groups may be coded. A filename may appear in more than one group.

Groups are defined by using the MCGROUP macro. The MCGROUP macro has three types: GROUP, FILE, and END.

GROUP

MCGROUP TYPE=GROUP,NAME=name

—defines the group name. Each time this macro appears, a new group is started.

FILE

MCGROUP TYPE=FILE,NAME=*name*

—adds a file name to the current group. This macro can appear up to 40 times per group. The file names can also be coded in list form. For example:

MCGROUP TYPE=FILE,NAME=(file1,file2,file3**,...)**

END

MCGROUP TYPE=END

—indicates the end of the group table. It must be coded after all other TYPE=GROUP and TYPE=FILE macros have been coded. The TYPE=END macro is mandatory.

Sample Table

A sample group table is located in source member AVBCICSG (AVBCICSG.A for VSE/ESA). This table must be assembled and linked into a sublibrary accessible to CICS using the job ASMCICSG (ASMCICSG.X for VSE/ESA).

Displaying AVB Operating Statistics

To display operating statistics for AVB

1. Execute the AVB1 transaction.
2. On the resulting AVB1 screen, type STATS on the Function line.
3. To display statistics for a specific file, type the file name on the Options line.
4. Then press ENTER.

   A set of report headings and execution statistics appears on the screen:

```
AVB V5.1.1 – OS/390 2.08 – CICS/TS 5.3
AVBridge V5.1.1 10/12/01                        10/16/01  17:00:54     AVB1A

Status:   AVB1000 – AVBridge Activated

Function:           ON/OFF/(STATS|FILES)/END
Options :           OPEN/CLOSE/VSAM/filename

READ    GETUPD  BROWSE  ADD     UPDATE  DELETE  ET      BT      ADA
000001  000001  000045  000021  000001  000021  000003  000000  000185
```

The columns display the following information:

Item	Indicates the number of . . .
READ	READ/GET commands issued
GETUPD	READ/GET commands issued with a HOLD placed on the record for update
BROWSE	READNEXT commands issued
ADD	records added to the files
UPDATE	records updated
DELETE	records deleted
ET	ETs issued
BT	BTs issued
ADA	calls to Adabas files

To display statistics for a specific file

☐ Enter the STATS request and specify a file name on the OPTIONS line.

Note:
The number of requests for the "STATS" display has a maximum usable upper value of 999,999. Use AVB Online Services to examine statistical values that exceed this range.

Displaying and Updating File Information

To display and update information about the locations and status of files

1. Execute the AVB1 transaction.
2. On the resulting AVB1 screen, type FILES on the Function line and press ENTER.

   AVB switches to the internal function UPDATE and a set of report headings and file information appears on the screen:

```
AVB V5.1.1 — OS/390 2.08 — CICS/TS 5.3
AVBridge V5.1.1 12/08/00                    12/15/00  17:49:46      AVB1A

Status:   AVB1000 — AVBridge Activated

Function: UPDATE    ON/OFF/(STATS|FILES)/END
Options :           OPEN/CLOSE/VSAM/filename

Database-ID:  00014

Filename I FNR   DBID  Filename I FNR   DBID  Filename I FNR   DBID

SYS001   O 00055 00014 SYS002   C 00055 00014 SYS003   V 00055 00014
SYS021   C 00056 00000 SYS21R   O 01055 00000 PLIESDS  O 00029 00235
```

3. (optional) Specify options for the FILES or UPDATE function.

   When the FILES function is specified and the Options line is left blank, the list of files starts from the first file of the transparency table.

   When the UPDATE function is active and the values on the screen are not modified, the next page of the transparency table is displayed each time you press ENTER.

When you specify a filename in the Options line, the file list starts from the specified file.

Other options that can be specified for the FILES or UPDATE function are

Option	Set the file status of all files to . . .
OPEN	“O”; open all files.
CLOSE	“C”; close all files
VSAM	“V”; forward all VSAM requests to VSAM.

The Database-ID field displays the global Adabas DBID.

The columns display the following information:

Item	Description
Filename	The VSAM DD name.
I	The file status (O, C, or V).
FNR	The number of the file in the Adabas database.
DBID	The Adabas database that contains the file.

The file status options have the following significance:

Status	Description
O	The file is opened and the calls are handled by Adabas.
C	The file is closed and the calls are rejected by AVB with NOTOPEN status.
V	The file is closed for AVB and the calls are forwarded to VSAM.

4. To modify the file status, the Adabas DBID, or the Adabas file number, overtype the corresponding values and press ENTER.

The same screen is displayed again showing the updated values.

Note:
More detailed file information is available using AVB Online Services.

Resident AVB Programs and Online Installations

Observe the following guidelines when performing online installations:

- Use CICS resource definition online (RDO) to load a new copy of a resident AVB module, such as the transparency table. Do **not** use CEMT NEW; if you do, the program is marked as nonresident, and results are unpredictable.
- Deactivate AVB before performing a CEDA installation for any group containing resident AVB programs.

9

AVB ONLINE SERVICES

This chapter tells you how to how to invoke AVB and run your VSAM application programs against Adabas databases using AVB online services, an online application written in Natural. Using AVB online services, you can

- activate and deactivate AVB;
- display, print, or download file definitions and statistics;
- modify the file status (open, closed, VSAM);
- display parameter setting of the AVB options table;
- analyze, print, or download the AVB trace table;
- display, print, or download the list of fixes applied to AVB;
- display product information about AVB online services; and
- maintain the global settings for the AVB online services session.

A built-in trace function and an interface to the Natural SYSRDC utility provide extensive debugging capabilities for AVB online services itself.

Prerequisites

AVB online services requires

- Natural version 2.3 or above available in the CICS where AVB is running for access.
- Natural Advanced Facilities for the “print“ function.
- Natural Connection for the “download” function.

Some AVB online services functions are also available using the appropriate AVB CICS transactions:

Transaction	Service
AVB1	Activate and deactivate AVB; display and modify the file status, display file statistics.
AVTR	Display AVB trace information and applied fixes.

Getting Started

Online Help

To activate the help feature

☐ Press PF1 on any AVB online services screen.

A general text describing the purpose and use of the currently selected screen is displayed.

To obtain help information for direct commands

☐ Issue the direct command HELP.

Function Keys

The following PF keys are defined throughout AVB online services:

PF Key	Value	Function
Enter	Refr	Refresh the information on the current screen
PF1	Help	Display help information
PF2	Flip	Show alternate PF-key settings (PF13 – PF24)
PF3	Exit	Exit from the current screen
PF4	<	Scroll to the left
PF5	>	Scroll to the right
PF7	Prev	Display the previous page
PF8	Next	Display the next page
PF10	– –	Display the first page
PF11	+ +	Display the last page
PF12	Menu	Display the AVB online services main menu
PF14	Flip	Show alternate PF-key settings (PF1 – PF12)

Command Line

The command line, which appears at the bottom of every AVB online services menu, allows you to enter commands that

- start certain AVB online services functions directly, even if the associated menu is not currently selected;
- select options from the main menu directly without regard to the menu currently being displayed; and
- execute the functions associated with PF keys.

To select main menu options

☐ Type the appropriate single-character code on the command line of any AVB online services menu.

See the section **Main Menu** on page 169 for a list of the single-character codes that you can enter as direct commands.

You can also enter the value associated with a PF key as a direct command.

For every direct command issued, an entry is inserted in the SYSRDC trace.

Examples:

- To display the AVB option table, type O on the direct command line of any AVB online services screen.
- To display the next page of the trace table, type NEXT on the direct command line in the trace table menu.

Direct Commands

The following direct commands are available:

Note:
Variable arguments are indicated in lowercase; an argument indicated in uppercase must be entered as provided.

Direct Command	Purposes
* comment	No action; “comment” is inserted in the SYSRDC trace.
AVBLOGO	Display the AVB logo screen.
FETCH prog	Call the external program “prog” using FETCH.
FETCH RETURN prog	Call the external program “prog” using FETCH RETURN.
FILE	List detailed information for the first file of the AVB transparency table.
FILE {n | name} {PAR | STA}	List the parameters or the statistical information for the n^{th} file of AVBTAB or the file named “name”.
FILELIST	List the files of AVBTAB.
FILELIST {PAR | STA}	List the parameters or the statistical information for the files of AVBTAB.
FIN	Exit from the Natural environment.
GLOBALS	Maintain the global settings in the user environment.
HELP	Display help information for the direct commands.
INFO	Display the AVB online services long error message corresponding to the most recently displayed AVB online services short error message.
INFO nnn	Display the AVB online services long error message corresponding to the AVB error number “nnn”.
LOGOFF	Exit from AVB online services.
LOGON lib	Log on to the Natural library “lib”.
MENU	Display the AVB online services main menu.
PRODUCT	Display product information for AVB such as the version, and the last correction applied.

Direct Command	Purposes
RDC	Display the SYSRDC main menu for debugging AVB online services. See the section **Tracing AVB Online Services** on page 191 for more information.
TRACE	Display the AVB online services trace menu. See the section **Tracing AVB Online Services** on page 191 for more information.
TRACE ?	Display the current status of the AVB online services trace mode.
TRACE COUNT	Reset the AVB online services trace counter.
TRACE NAME prog	Restrict AVB online services tracing to the program “prog”. If “prog” is omitted or set to ALL, all programs are traced.
TRACE OFF	Stop the AVB online services trace.
TRACE ON	Start the AVB online services trace.
TRACE nn	Start the AVB trace type “nn”. See the section **Tracing AVB Online Services** on page 191 for more information.
VERSION	Display product information for AVB itself such as the version and the last correction applied.
ZAPS	Display the AVB ZAP table.

Preparing the User Environment

To meet the needs of your environment, you can edit source member PROFILES in the SYSAVB library to

- add separate profiles for individual users; or
- adjust default values for some AVB variables.

The following variables can be set in the profile:

Glabal Variable	Value	Description
#DATE_FORMAT	E	Date format is DD/MM/YYYY
#DATE_FORMAT	G	Date format is DD.MM.YYYY
#DATE_FORMAT	I	Date format isYYYY–MM–DD
#DATE_FORMAT	U	Date format is MM/DD/YYYY
#PRINTER_NAME	pr_name	Name of a printer
#LOGO	Y	The AVB logo is displayed when AVB online services starts.
#LOGO	N	The AVB logo is not displayed when AVB online services starts.

#DATE_FORMAT selects the display format for the date. The format selected is used on all AVB online services screens.

#PRINTER_NAME specifies the name of the default printer. For any print request, you can overwrite this default printer name in a pop-up window. If you specify the printer name “SCREEN”, the output is routed to the screen.

#LOGO specifies whether the AVB logo is displayed when AVB online services starts. You can view the AVB logo from any AVB screen by entering the direct command “AVBLOGO“.

The values of these variables can be modified during an AVB session as described in the section **Maintain the Global Settings** on page 182.

Accessing the Main Menu

To enter AVB online services

☐ Log on to the Natural application SYSAVB and start the program MENU.

```
12:14:38                *** AVB ONLINE SERVICES ***              2001-09-25
AVB Active                     - Main Menu -                     MAVB0001
                                                                 DBID 14


                Code                Service
                ____  ____________________________________
                 A    Deactivate AVB
                 F    File Menu
                 G    Global Settings
                 O    Options Table
                 T    Trace Table
                 Z    Zaps and Product Information
                 ?    Help
                 .    Terminate
                ____  ____________________________________
          Code .. _


     Direct command ==> _____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit                                                  Menu
```

The main menu lists the available functions. Like most AVB Online System screens, it shows the status of AVB as “Active”, “not Active”, or “not Installed”.

The main menu function “Trace Table” is only available when AVB is active. If you receive the message “AVB is not installed” when you start AVB online services, AVB has not been installed properly. Verify the AVB installation in the CICS environment as described in the chapters **OS/390 Installation** or **VSE/ESA Installation**, as appropriate.

The DBID displayed in the main menu and in most AVB screens is the global Adabas DBID for the AVB session as defined in the AVB transparency table.

To terminate the AVB online services session

☐ Press PF3 on the main menu or select the code ".".

Alternatively, you can terminate the AVB online services session from any screen by issuing the direct command FIN or LOGOFF.

Main Menu Functions

The following services are available from the main menu:

Code	Service	Page
A	Activate or deactivate AVB. Additional VSAM calls will either go to VSAM (active) or not (not active).	171
F	Display, print, or download the AVB file information as defined in the transparency table. You can also change the status of selected files or all files.	172
G	View and modify the global settings for your current AVB online services session.	182
O	Display, print, or download the AVB system dependencies and user options as defined in the AVB options table.	183
T	Analyze, print, or download the entries of the AVB trace table as collected during the current AVB session.	184
Z	Display, print, or download the AVB ZAP status and product information.	188

To select a main menu service

☐ Type the appropriate one-character code on the direct command line from any AVB online services screen.

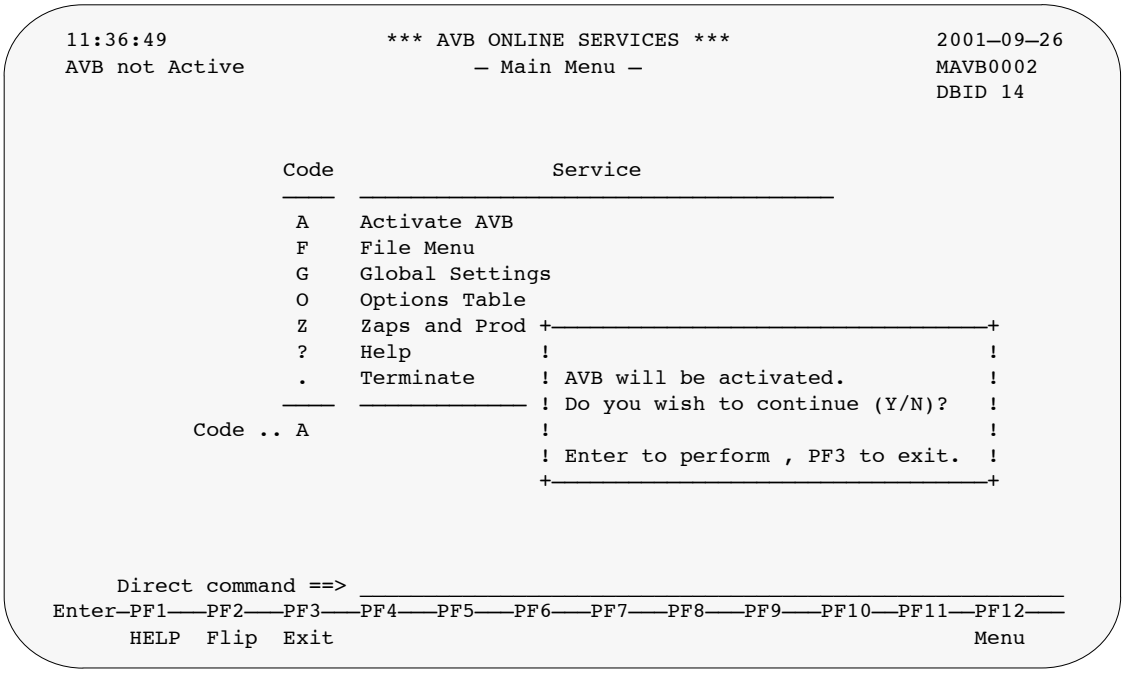
Subsequent sections in this chapter describe the functions of AVB online services and the menu/screen structures in the order they appear on the main menu.

Activate or Deactivate AVB

Depending on the current status of AVB, you can activate or deactivate AVB by either

- selecting the code A on the main menu; or
- entering A in any direct command line.

A window is displayed:



If you enter a Y in the window, the selected function is executed.

File Menu

You can access the file menu either by

- selecting code F on the main menu; or
- entering F on any direct command line.

The file menu lists functions to display, print, or download AVB file information as defined in the transparency table. When AVB is active, you can change the status of the files.

```
11:49:16                  *** AVB ONLINE SERVICES ***                2001-09-26
AVB Active                       - File Menu -                       MFMENU01
Total Files: 80                                                      DBID 14


                        Code  Service
                        ____  _________________________
                          C   Close file(s)
                          D   Download
                          M   MCTAB Parameters
                          O   Open file(s)
                          P   Print
                          S   Statistical Information
                          V   VSAM file(s)
                          .   Exit
                        ____  _________________________
                   Code .. _

      File Name/Number ... ________
      File2 Name/Number .. ________ (for Print/Download)

     Direct command ==> ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit                                                   Menu
```

The total number of files defined in the transparency table is provided on the file menu and on all screens accessed by the file menu.

Services Available from the File Menu

The following table shows the services available from the file Menu.

Note:
The services C, O, and V are only available when AVB is active.

The File and File2 columns indicate whether the fields File Name/Number or File2 Name/Number are filled.

You can specify in these fields either a VSAM file name or a file number. The file number corresponds to the sequence in the transparency table; for example, the first file in the transparency table has the number 1, the second has the number 2.

Code	File	File2	Service
C	N	N	Close all files.
C	Y	N	Close the specified file.
D	{Y | N}	{Y | N}	Download file parameters and statistical information. The download starts at the file specified in File Name/Number and ends at the file specified in File Name/Number 2. If the File / File2 field is left empty, the download starts from the first file / ends at the last file, respectively.
M	N	N	Display the MCTAB parameters of all files.
M	Y	Y	Display the MCTAB parameters of all files. The list starts at the file specified in File Name/Number, but you can page backward and forward through all files. File2 is ignored.
M	Y	N	Display the MCTAB parameters for the specified file.
O	N	N	Open all files.
O	Y	N	Open the specified file.
P	{Y | N}	{Y | N}	Print file parameters and statistical information. The print starts at the file specified in File Name/Number and ends at the file specified in File Name/Number 2. If the File / File2 field is left empty, the print starts from the first file / ends at the last file, respectively.
S	N	N	Display statistical information of all files.

Code	File	File2	Service
S	Y	Y	Display statistical information for all files. The list starts at the file specified in File Name/Number, but you can page backward and forward through all files. File2 is ignored.
S	Y	N	Display statistical information for the specified file.
V	N	N	Forward all VSAM requests to VSAM.
V	Y	N	Forward VSAM requests against the specified file to VSAM.

If you change the status of any or all files, you need to confirm the action in a window. More information about the status of the files can be found in the chapter **CICS Operation**, section **Opening and Closing Files** on page 155.

The screens accessed by the functions M or S are described in the following sections.

MCTAB Parameter List

To access the MCTAB parameter list

☐ Do one of the following:

- select code M in the file menu;
- press PF9 on the statistical information screen; or
- enter the command FILELIST PAR on any direct command line.

The parameters of the files as defined in the AVB transparency table macro MCTAB are displayed. From this list you can display detailed file information or change the status of selected files.

```
17:13:42                  *** AVB ONLINE SERVICES ***                2001-09-27
AVB Active                   - MCTAB  Parameters -                   MFLISTP1
Page 1/4                       Total Files: 80                       DBID 14

Cmd   Nbr File     Status   DBID FDSN-File-Dataset-Name                    FNR
 _      1 SYS001   Open       14 AVB.SYS001.D0000.F055                     55
 _      2 SYS002   Closed     14 AVB.SYS002.D0000.F055                     55
 _      3 SYS003   VSAM       14 AVB.SYS003.D0000.F055                     55
 _      4 SYS004   Open       14 AVB.SYS004.D0000.F055                     55
 _      5 SYS005   Open       14 AVB.SYS005.D0000.F055                     55
 _      6 SYS006   Open       14 AVB.SYS006.D0000.F055                     55
 _      7 SYS007   Open       14 AVB.SYS007.D0000.F055                     55
 _      8 SYS008   Open       14 AVB.SYS008.D0000.F055                     55
 _      9 SYS009   Open       14 AVB.SYS009.D0000.F055                     55
 _     10 SYS010   Open       14 AVB.SYS010.D0000.F055                     55
 _     11 SYS011   Open       14 AVB.SYS011.D0000.F055                     55
 _     12 SYS012   Open       14 AVB.SYS012.D0000.F055                     55
 _     13 SYS013   Open       14 AVB.SYS013.D0000.F055                     55
 _     14 SYS014   Open       14 AVB.SYS014.D0000.F055                     55

    Direct command ==> _____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit  <     >          Prev  Next  Stats --    ++    Menu
```

Each parameter is described in detail in the detailed parameter list, which is displayed when you enter M on the line command for any file. The MCTAB parameters are also described in the chapter **Creating the Transparency Table** starting on page 95.

The MCTAB parameters are sorted alphabetically.

To display more MCTAB parameters

☐ Scroll to the right with PF5.

To start the list from the given file number or file name

☐ Overwrite the “Nbr” or “File” field, respectively.

The following line commands are available:

Cmd	Description
M	Display the MCTAB parameters for the file.
S	Display detailed statistical information for the file.
O	Open the file.
C	Close the file.
V	Set the file status to “VSAM”; that is, forward the VSAM requests against the file to VSAM.

Special Function Key Settings

PF Key	Value	Function
PF9	Stats	Display the statistical information of all files.
PF16	Open	Open all files.
PF17	Close	Close all files.
PF18	VSAM	Set the status of all files to “VSAM”.

Detailed Parameter List

To access the detailed parameter list

☐ Do one of the following:

- specify code M and a file name or number in the file menu;
- enter the line command M in the MCTAB parameter list;
- enter the line command M in the statistical information screen;
- press PF9 in the detailed statistical information screen; or
- enter the direct command “FILE {n | name} PAR” on any screen.

The list displays the parameters of the file and a detailed description of each.

```
11:53:48                *** AVB ONLINE SERVICES ***                2001-09-28
AVB Active             - Parameters of File SYS021   -             MFILEP01
Page 1/2                     File 21    of 80                      DBID 14

Parameter Description                                              Value
--------- -------------------------------------------------------- --------------------
(OCIND)   Current file status ........................... Open
DBID      Adabas DBID of the file ....................... 14
FDSN      File dataset name ..... AVB.SYS021.D0000.F055
FNR       Adabas file number ............................ 55
GLBLFID   Global format Id .............................. DYNAMIC
INDXTYP   KEY1=primary (P) or secondary (S) key ......... P
KEY1      Primary/secondary key name ..................... SK
KEYLEN    Length of the key field ........................ 7
KEYOFF    Offset of the key .............................. 3
LAxxxx    Number of long alpha (LA) fields ............... 0
ODxxxx    Number of occurs-depending-on (ODO) fields ..... 3
PCKKEY    Format of the key field is packed .............. N


    Direct command ==> ____________________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Flip  Exit  -File +File       Prev  Next  Stats --F   ++F   Menu
```

The parameters are sorted alphabetically.

To display more MCTAB parameters

☐ Scroll down by pressing PF8.

Note:
The OCIND parameter reflects the current status of the file rather than the originally specified OCIND value.

Special Function Key Settings

PF Key	Value	Display . . .
PF4	–File	the parameters of the previous file in the transparency table.
PF5	+File	the parameters of the next file in the transparency table.
PF9	Stats	detailed statistical information for the current file.
PF10	—F	the parameters of the first file in the transparency table.
PF11	++F	the parameters of the last file in the transparency table.

Statistical Information

To access the statistical information screen

☐ Do one of the following:

- select code S in the file menu;
- press PF9 in the MCTAB parameter list; or
- enter the command “FILELIST STA” on any direct command line.

Statistical information is displayed for the AVB files collected in the current AVB session. You may additionally display detailed statistical information or change the status of selected files.

```
16:25:15                *** AVB ONLINE SERVICES ***                2001-09-28
AVB Active              - Statistical Information -                MFLISTS1
Page 1/4                     Total Files: 80                       DBID 14

Cmd   Nbr File     Status        READ READ-UPDATE     BROWSE        ADD
 _     15 SYS015   Open             0           0          0          0
 _     16 SYS016   Open             0           0          0          0
 _     17 SYS017   Open             0           0          0          0
 _     18 SYS018   Open             0           0          0          0
 _     19 SYS019   Open             0           0          0          0
 _     20 SYS020   Open             0           0          0          0
 _     21 SYS021   Open             1           1         45         21
 _     22 SYS022   Open             0           0          0          0
 _     23 SYS023   Open             0           0          0          0
 _     24 SYS024   Open             0           0          0          0
 _     25 SYS025   Open             0           0          0          0
 _     26 SYS026   Open             0           0          0          0
 _     27 SYS027   Open             0           0          0          0
 _     28 SYS028   Open             0           0          0          0

    Direct command ==> _____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit  <     >          Prev  Next  Parms --    ++    Menu
```

To display detailed statistical information

☐ Enter the line command S for any file.

The detailed statistical information describes each statistical value in the same sequence.

To start the list from the given file number or file name

☐ Overwrite the “Nbr” or “File” field, respectively.

The following line commands are available:

Cmd	Description
M	Display the MCTAB parameters for the file.
S	Display detailed statistical information for the file.
O	Open the file.
C	Close the file.
V	Set the file status to “VSAM”; that is, forward the VSAM requests against the file to VSAM.

Special Function Key Settings

PF Key	Value	Function
PF9	Parms	Display the MCTAB parameters of all files.
PF16	Open	Open all files.
PF17	Close	Close all files.
PF18	VSAM	Set the status of all files to “VSAM”.

Detailed Statistical Information

To access detailed statistical information

☐ Do one of the following:

- specify code S and a file name or number on the file menu;
- enter the line command S in the MCTAB parameter list;
- enter the line command S on the statistical information screen;
- press PF9 in the detailed parameter list; or
- use the direct command “FILE {n | name} STA” on any screen.

Detailed statistical information is displayed for the file.

```
16:27:34                *** AVB ONLINE SERVICES ***                2001-09-28
AVB Active             - Statistics of File SYS021   -             MFILES01
Page 1/2                     File 21    of 80                      DBID 14

Description                                                      Value
_____________________________________________________________ ____________________
Current file status ..................................... Open
Number of READ commands ..................................          1
Number of READ UPDATE commands ...........................          1
Number of BROWSE commands ................................         45
Number of ADD commands ...................................         21
Number of UPDATE commands ................................          1
Number of DELETE commands ................................         21
Number of SPP USER/KC commands ...........................          3
Number of SPP ROLLBACK commands ..........................          0
Number of Adabas calls ...................................        108
Total number of Adabas calls (all files) .................        185
Number of active tasks using the file ....................          0


     Direct command ==> _____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit  -File +File       Prev  Next  Parms --F   ++F   Menu
```

To display more statistical information

☐ Scroll down by pressing PF8.

Special Function Key Settings

PF Key	Value	Display . . .
PF4	–File	statistical information for the previous file in the transparency table.
PF5	+File	statistical information for the next file in the transparency table.
PF9	Parms	the MCTAB parameters of the current file.
PF10	—F	statistical information for the first file in the transparency table.
PF11	++F	statistical information for the last file in the transparency table.

Maintain the Global Settings

To access the global settings menu

☐ Do one of the following:

- select code G on the main menu; or
- enter the command G on any direct command line.

From this menu, you can display and modify the global settings for your current AVB online services session.

```
17:22:32                 *** AVB ONLINE SERVICES ***               2001-09-28
User LHU1                    - Global Settings -                   MGLOB001

                          Format Area          Date Format
                          ------ ------------- ----------
                             E   Europe        DD/MM/YYYY
                             G   Germany       DD.MM.YYYY
                             I   International YYYY-MM-DD
                             U   USA           MM/DD/YYYY
                          ------ ------------- ----------
Date Format ............. I

Display LOGO Screen (Y/N) .. Y

Printer ................. SCREEN__




    Direct command ==> ____________________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Flip  Exit        Save                                      Menu
```

To modify the global settings

☐ Overwrite the values and save your changes by pressing PF5.

Such modifications are retained only for the current AVB online services session. If you exit and reenter AVB online services, these modifications are lost and the original default values are reinstated.

To permanently change the global settings

☐ Edit the source program PROFILES as described in the section **Preparing the User Environment** on page 168.

This section also provides a detailed description of the variables.

Options Table

To access the options table screen

☐ Do one of the following:

- select code O on the main menu; or
- enter the command O on any direct command line.

From this screen, you can display, print, or download the AVB system dependencies and user options as defined in the AVB options table.

```
18:01:17                *** AVB ONLINE SERVICES ***              2001-09-28
AVB Active                   - Options Table -                   MOPT0001
Page 1/2                                                         DBID 14

Parameter Description                                          Value
--------- -------------------------------------------------- --------------------
ADABNAM   Name of the batch Adabas link routine ......... ADALNK
ADALNAM   Name of the CICS Adabas link routine .......... ADABAS
ADATNAM   Name of the Adabas CICS task-related user exit  ADATRUE
ASYNCL    AVB TRUE issues CL when invoked asynchronous... NO
ADAVER    Adabas version number ......................... 7
COBPARM   Pass system parameters (batch) ................ YES
CTRDMP    Generate snapshot dumps ....................... NO
CUST      Customer name or ID ...........................
DBGNZO    Save only nonzero rsp codes in debug trace .... NO
DYNMGFID  Support for the dynamic global format ID ...... ON
EOTRC     Issue RC at end of task ....................... NO
ETRC      Issue RC with each ET or BT command ........... YES
INITRC    Issue RC at the start of each transaction ..... NO


    Direct command ==> ____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit        Print PC    Prev  Next        --    ++    Menu
```

The parameters are listed with a detailed description. They are also described in the chapter **Creating the Options Table** starting on page 133.

The MCTAB parameters are sorted alphabetically.

To display more MCTAB parameters

☐ Scroll down with PF8.

Special Function Key Settings

PF Key	Value	Function
PF5	Print	Print the AVB options table.
PF6	PC	Download the AVB options table.

Trace Table

To access the trace table screen

☐ Do one of the following:

- Select code T on the main menu; or
- Enter the command T on any direct command line.

From this screen, you can analyze, print, or download the AVB trace table.

```
17:21:45                 *** AVB ONLINE SERVICES ***               2001-10-01
AVB Active                     - Trace Table -                     MDEBUG01
Timestamp 17:18:28               399 Entries                       DBID 14

Nbr   Counter  Funct. Cmd Option   File       FNR VS Rsp K-Len        ISN Term
100         100  GET NE L3         SYS021      55 00   0     0        573 0814
101         101  GET NE L3         SYS021      55 00   0     0        574 0814
102         102  GET NE L3         SYS021      55 00   0     0        575 0814
103         103  GET NE L3         SYS021      55 10   3     0        575 0814
104         104  FIND   RC EQGT     SYS021      55      0     0          0 0814
105         105  FIND   L9 EQGT     SYS021      55 00   0     0          0 0814
106X        106  GET NE RC         SYS021      55      0     0          0 0814
107         107  GET NE L3         SYS021      55 00   0     0        555 0814
108         108  DELETE E4         SYS021      55 00   0     0        555 0814
109         109  GET NE L3         SYS021      55 00   0     0        556 0814
110         110  DELETE E4         SYS021      55 00   0     0        556 0814
111         111  GET NE L3         SYS021      55 00   0     0        557 0814
112         112Y DELETE E4         SYS021      55 00   0     0        557 0814
113         113  GET NE L3         SYS021      55 00   0     0        558 0814

    Direct command ==> _____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Refr  HELP  Flip  Exit  <     >     PC    Prev  Next  Print ---    ++    Menu
```

The AVB trace table contains one entry per Adabas request. The Adabas calls listed in the table were issued by AVB during the current AVB session either to fulfill VSAM requests or to satisfy internal requirements. The table wraps around; it always contains the most current trace entries.

When you enter the trace table screen, a snapshot is taken of the full AVB trace table. The data captured in the snapshot is not updated while you analyze, print, or download it.

To refresh the snapshot data

☐ Press ENTER with no additional input.

The time when the snapshot is taken (”Timestamp”) and the maximum number of entries in the trace table are displayed at the top of the screen.

Interpreting the Trace Table Display

The columns in the trace table display the following information:

Column	Description
Nbr	Number of the entry in the table The entry number is always in the range from 1 to the maximum number of entries in the table. For technical reasons, the maximum number of entries displayed in this column is one less than the value specified in the options table.
Counter	Adabas call counter This is stored in a 3-byte field. If the maximum count of 16.777.215 is reached, the counter is reset to 0.
Funct	Internal AVB function code
Cmd	Generated Adabas command
Option	Internal AVB option code
File	File name
FNR	Adabas file number
VS	VSAM status code (empty for AVB internal Adabas calls)
Rsp	Adabas response code
K–Len	Key length
ISN	Adabas ISN value
Term	Terminal ID originating the task
IBL	ISN buffer length
RBL	Record buffer length
Additions-5	Adabas additions 5 field in character and hexadecimal format
TCAFCTR	CICS request code
Prefetch	Command option 1 when prefetch is active

Modifying the Trace Table Display

You can modify the trace table display by overwriting the title of a column:

To display a specific row

☐ Overwrite “Nbr” with an existing entry number or overwrite “Counter” with an existing counter value.

To mark specific rows for later display

☐ Position the row containing the “Nbr” or “Counter” column value to be marked with X or Y as the first in the table display.

☐ Overwrite the relevant column title with .X or .Y, respectively.

Notes:

1. *The marked rows in the “Nbr” column are used as default values for the print range.*
2. *Once the trace table is full, new entries force old entries off the display so that a “Counter” value becomes associated with a different “Nbr” value, even though the “marks” remain with the values originally marked.*

To display the marked row later

☐ Specify on the command line either FX or FY as appropriate.

To search for and display the next row after the first that contains a specific value

☐ Overwrite the column title “Funct”, “Cmd”, “File”, “FNR”, “VS”, “Rsp”, “ISN”, or “Term”, as appropriate, with the value.

For the columns “Funct”, “Cmd”, “File”, and “Term”, it is sufficient to specify just the relevant part of the value; for example, if you specify “S02” in the “File” column, the next file name containing an “S02” (perhaps SYS020) is displayed.

To search for and display the next row after the first that contains the same value as the first

☐ Overwrite the column title “Funct”, “Cmd”, “File”, “FNR”, “VS”, “Rsp”, “ISN”, or “Term”, as appropriate, with an equals sign (=).

To display the next nonzero VSAM status code or Adabas response code

☐ Overwrite “VS” or “Rsp”, respectively, with an “X”.

Special Function Key Settings

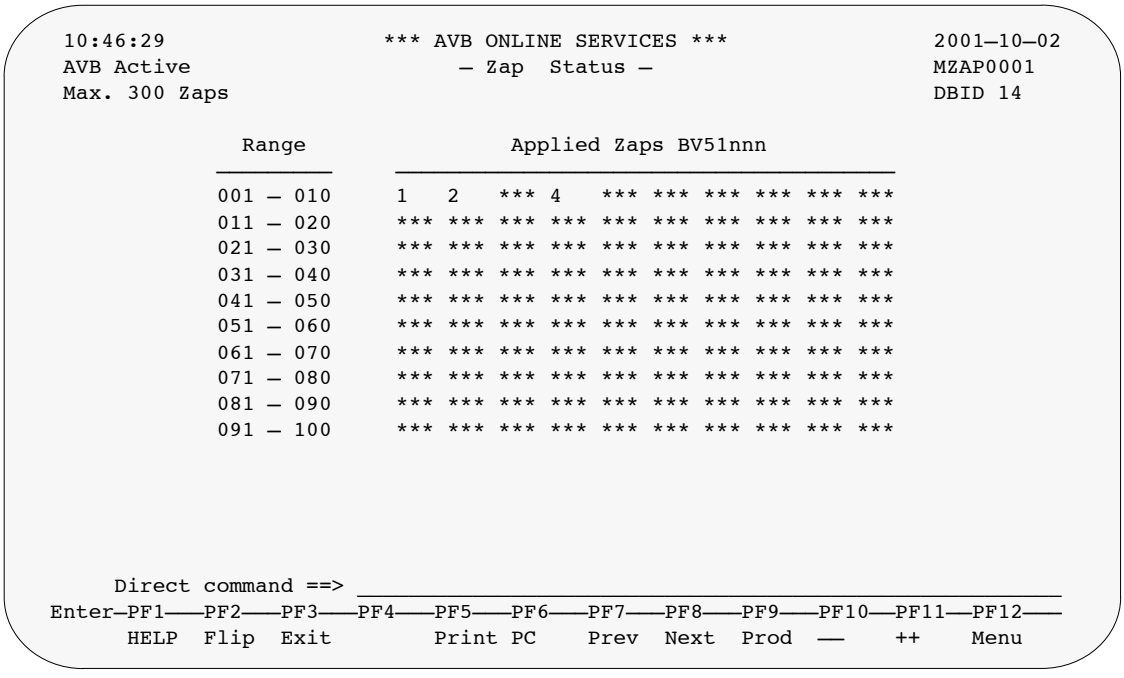
PF Key	Value	Function
PF6	PC	Download the full AVB trace table.
PF9	Print	Print the AVB trace table. The columns currently displayed are printed. If you want to print other columns, scroll to the right or left and press PF9 again. You can specify a range or rows to print in a window. If any rows in the “Nbr” column have been marked with “X” or “Y”, these values are used to define the default print range. Otherwise, the first and last rows are used to define the default print range. Empty rows are not printed.

ZAPs and Product Information

Displaying the AVB ZAP Status

To access the ZAP Status screen

☐ Select code Z on the main menu or enter Z on any direct command line.



You can display, print, or download the list of fixes (ZAPs) applied to AVB.

The AVB ZAP table contain one entry per ZAP. If a ZAP is applied, the corresponding number is displayed; otherwise, the entry is shown as “***”.

The maximum possible number of ZAPs is displayed at the top of the screen.

To start the list from a particular ZAP number

☐ Overwrite the first “Range” value with a valid number.

Special Function Key Settings

PF Key	Value	Function
PF5	Print	Print the AVB ZAP table.
PF6	PC	Download the AVB ZAP table together with the AVB online services product information.
PF9	Prod	Display the AVB online services product information.

Displaying the AVB Online Services Product Information

To access the Product Information screen

☐ Press PF9 on the ZAP Status screen or enter PRODUCT or VERSION on any direct command line.

```
12:33:33                *** AVB ONLINE SERVICES ***              2001-10-02
User LHU1                 - Product Information -                MVERSION

Product .............: Adabas Bridge for VSAM (AVB)
Utility .............: AVB Online Services (AVB OS)
AVB OS version ......: 5.1.1
Freeze date .........: 2001-mm-dd
Last correction ....: BV51nnn
Catalog date .......: 2001-mm-dd
AVB nucleus version : 5.1.1
Natural version ....: 23
CICS Level..........: 0530









     Direct command ==> _____________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Flip  Exit              PC                Zaps              Menu
```

You can display or download information about AVB Online Services and some related products.

The Product Information screen displays the version and the freeze date of AVB online services together with the name and catalog date of the last correction applied. Because AVB online services corrections are cumulative, all those with numbers lower than the one displayed are also applied. In addition, the version of the AVB nucleus, the Natural version, and the CICS level are displayed.

Special Function Key Settings

PF Key	Value	Function
PF6	PC	Download the AVB Online System product information together with the AVB ZAP table.
PF9	Zaps	Display the AVB ZAP table.

Tracing AVB Online Services

Two utilities are provided for debugging AVB online services itself:

- the built-in AVB online services internal trace function; and
- an interface to the Natural SYSRDC utility.

Internal Trace Function

The AVB online services internal trace facility currently offers the following trace "types":

Type	Displays . . .
Program trace	the name of each Natural object executed, as well as a counter, the object type (for example, "program"), and the level.
General trace information	a variety of information about the functions that are currently executing.
AVBCICS calls	the calling parameters and the results received from calls to the AVB nucleus program AVBCICS.

To access the Trace Maintenance screen

☐ Enter TRACE on any direct command line.

```
14:40:06                *** AVB ONLINE SERVICES ***                2001-10-02
User LHU1                  - Trace Maintenance -                   MTRACE01

Mk  Nr Description                       Mk  Nr Description
--- -- --------------------------------  --- -- --------------------------------
 _   1 Program trace                       _   9 N/A
 _   2 General trace information           _  10 N/A
 _   3 AVBCICS calls                       _  11 N/A
 _   4 N/A                                 _  12 N/A
 _   5 N/A                                 _  13 N/A
 _   6 N/A                                 _  14 N/A
 _   7 N/A                                 _  15 N/A
 _   8 N/A                                 _  16 N/A

Program to be traced .. ALL
Program counter ....... 0000000



    Direct command ==> ____________________________________________________________
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      HELP  Flip  Exit        Selct                         Off   On    Menu
```

The Trace Maintenance screen displays the available trace types. From this screen, you can start or stop specific trace functions and set trace-specific values.

To activate one or more trace types

☐ Mark them with “X”.

☐ Press PF5 before you leave the screen

When the AVB online services internal trace has been activated, the trace information is written to the screen.

To page through the trace information

☐ Press ENTER until you reach the next AVB online services screen.

To deactivate one or more trace types

☐ Remove the X.

Program to be Traced

To restrict tracing to a specific AVB online services program

☐ Specify the program name in the field “Program to be traced”.

To trace all AVB online services programs (the default)

☐ Specify ALL in the field “Program to be traced”.

Program Counter

The counter field, which is written out together with the program trace, counts the number of Natural objects traced by AVB online services.

To set the counter

☐ Overwrite the field “Program counter” with any value.

Direct Commands

Most functions that can be executed from the Trace Maintenance screen can also be executed using direct commands. The direct commands are described in the section **Direct Command Line** on page 166.

Special Function Key Settings

PF Key	Value	Function
PF5	Selct	Select the current trace type settings.
PF10	Off	Deactivate all trace types.
PF11	On	Activate all trace types.

Interface to the Natural SYSRDC Utility

The Natural data collection utility SYSRDC supplies monitoring and accounting data for certain events within Natural. The SYSRDC utility is described in detail in the *Natural Utilities Manual for Mainframes*. Refer to that manual for any question regarding the installation or activation of SYSRDC, or the meaning of the various event types and data fields collected by SYSRDC.

The SYSRDC trace buffer records the data in wrap-around mode. You will therefore find only the last events in the buffer. The number of events that fit in the buffer depends on the setting of the corresponding Natural parameter.

Some services are provided for monitoring AVB online services. The programs corresponding to these services are used as an interface to SYSRDC and are delivered on the SYSAVB library.

To access the services provided for monitoring AVB online services

☐ Issue the direct command RDC from any AVB online services screen.

```
17:58:26           *** NATURAL  S Y S R D C  UTILITY ***           2001—10—02
User LHU1                        — Main Menu —                     MRDC0001

                   Code                Service
                  ──────  ──────────────────────────────────────
                    S     Start SYSRDC trace (clear buffer)
                    P     Stop SYSRDC trace recording
                    E     Event selection
                    L     List SYSRDC buffer (stop trace)
                    ?     Help
                    .     Terminate
                  ──────  ──────────────────────────────────────
       Enter code : _

    Current state : SYSRDC trace stopped

   List from line : 1      to line : 999999
           Screen : X  Printer : _  PC—File : _


     Direct command ==> ____________________________________________________
Enter—PF1——PF2——PF3——PF4——PF5——PF6——PF7——PF8——PF9——PF10—PF11—PF12——
      HELP  Flip  Exit                                                Menu
```

The following services are available from the SYSRDC main menu:

Code	Service Description
S	Start the SYSRDC data collection. Before the data collection is started, the SYSRDC buffer is cleared automatically. You need to confirm this action in a window.
P	Stop the SYSRDC data collection. You need to confirm this action in a window.
E	Display the SYSRDC Event Selection screen, from which you can select the events to be collected or listed.
L	List the content of the SYSRDC buffer. Before the buffer is listed, the data collection is stopped automatically. You need to confirm this action in a window. The output is routed to the screen, to a printer, or downloaded to a PC if you mark the corresponding field with an X. You can specify a “from line” and a “to line” value to restrict the lines listed.

10

BATCH OPERATION

After you have migrated your VSAM application to Adabas, you can invoke AVB and run your VSAM application programs against Adabas databases.

This chapter discusses the batch operation of AVB. It provides a detailed discussion of AVB logic: how AVB is activated, how it processes requests, and how it is deactivated.

Activating / Deactivating AVB

OS/390 or z/OS

In batch, every job step that requires AVB must execute AVBLOAD, which

- obtains storage for use by AVB and loads resident AVB modules;
- establishes an SVC screen for OPEN/CLOSE; and
- loads and executes the application program.

Modify the JCL to invoke AVBLOAD. Include AVBUSER and ADARUN parameter statements in the JCL. These statements are discussed in this chapter.

When the application terminates, the AVB resource manager is invoked to deactivate SVC screening of OPEN and CLOSE in the address space. In addition, the resource manager produces end-of-jobstep AVB statistics and frees common storage used by AVB during jobstep execution.

VSE/ESA

AVB may be activated during the IPL by adding

- SET SDL statements and execution steps for AVBINST; and
- AVBSWI to the ASI PROC.

Alternatively, for testing or manual activation of AVB in a VSE/ESA system, the job AVBACT.X may be modified and executed. This job must be run in the background partition on VSE/ESA systems below version 2.6.

The SET SDL statements place the AVB resident $$B transients, the job control exit, partition vector table, and the AVBAVBO2 open subroutine in the SDL.

The AVBINST program is then run to install the AVB job exit. This may be installed as the phase AVBJBXT, or as a phase named in the system job exit table. For more information on the AVB job exit program and the use of the job exit table in VSE/ESA, see section .....

The program AVBSWI, which is used to activate AVB during the IPL, acts as an ON/OFF switch to activate and deactivate AVB. If AVB is inactive, AVBSWI activates it; if AVB is active, AVBSWI deactivates it. AVBSWI writes a message to the console indicating whether it is activating or deactivating AVB.

If SYSLST is properly assigned during the IPL, errors detected by the AVB job exit are reported in the printout of the statement on SYSLST in column 29 of the card being printed. If SYSLST is not properly assigned, only console messages are written by AVB.

During the OPEN call, AVB determines whether to direct a file to Adabas by matching the file name (DTF-name) in the file access control block (ACB) with the entry in the transparency table. AVB then opens the Adabas file and updates the ACB to direct all subsequent requests for the file to the AVB module AVBRIDGE.

Note:
When a // JOB statement is processed by job control, the partition table that contains information about the following is cleared: the status of AVB (ON/OFF), the transparency table to be used, the files to be opened as VSAM files, and additional information required by AVB.

Batch Processing

OPEN and CLOSE Requests

OS/390 or z/OS

Under OS/390 (z/OS), AVB processes OPEN requests in batch in the following steps:

1. An application program issues an OPEN request.
   - If AVB is inactive in the batch address space, the request is sent immediately to the operating system's OPEN routines.
   - If AVB is active in the batch address space, the OPEN or CLOSE request from the application is intercepted by AVB's SVC screening routine AVBSCRN.
2. AVBSCRN determines whether the addresses of the passed parameters point to VSAM ACBs and whether the DD names in the ACBs match entries in the AVB transparency table.
   - If a match is found, the file is opened for AVB processing.
   - If the parameter address points to a DCB, or the DD name in the VSAM ACB is not found in the AVB transparency table, the OPEN request is passed to the operating system's standard OPEN routines.

   When AVBSCRN intercepts a CLOSE request, it determines whether the address in the ACBINRTN field in the VSAM ACB matches that of AVBHOOKR. If the addresses do not match, AVBSCRN passes control to the operating system.

   If the addresses match, AVBSCRN calls the AVBHOOKC routine, which removes the AVBHOOKR address from the ACB and restores the ACB to the state it was in before AVB processed the OPEN. AVBHOOKC then invokes the AVB nucleus, which converts the request to an Adabas command and calls Adabas to close the Adabas file.

VSE/ESA

Under VSE/ESA, AVB is activated for a job step by the presence of * AVBUSER cards in the JCS. These are detected and processed by the AVB job control exit. The job control exit initializes some storage and sets up the partition for AVB processing.

The application program is then run from the EXEC card of the job step JCS. When an OPEN request is encountered, the AVB $$B OPEN transient ($$AVBO1) is invoked by the system's generalized VSAM open routines.

The DTF-name in the VSAM ACB is compared to entries in the AVB transparency table:

- If a match is found, the address of the AVB general I/O routine is placed in the VSAM ACB so that all subsequent requests against the file are processed by Adabas Bridge for VSAM until the file is closed.
- If a match is not found, control passes to the operating system's VSAM open routines for handling.

A CLOSE request is routed to the AVB $$B CLOSE transient ($$BAVBC1) by the general VSAM close routine. It determines whether the address in the ACBINRTN field in the VSAM ACB matches that of AVBHOOKR.

- If the addresses match, the $$B CLOSE transient calls the AVBHOOKC routine, which removes the AVBHOOKR address from the ACB and restores the ACB to the state it was in before AVB processed the OPEN. AVBHOOKC then invokes the AVB nucleus, which converts the request to an Adabas command and calls Adabas to close the Adabas file.
- If the addresses do not match, the $$B CLOSE transient passes control to the operating system.

Process Illustration

This process is illustrated in the following figure:

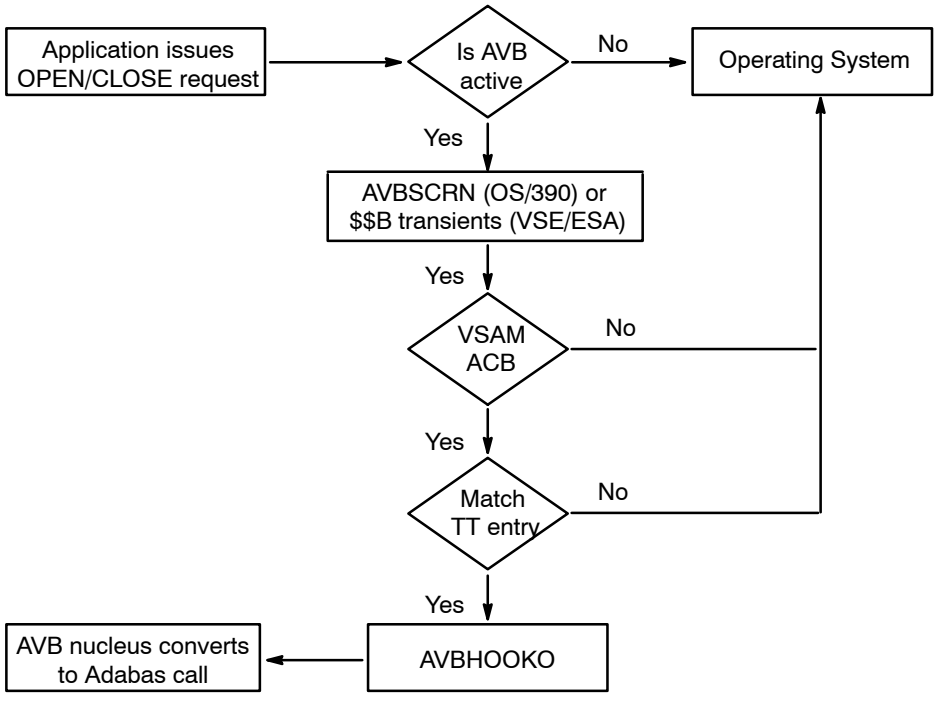


Figure 10-1: AVB Batch Processing of OPEN/CLOSE Requests

READ and UPDATE Requests

During OPEN processing of a bridged file (a file with an entry in the AVB transparency table), AVB places the address of its own I/O handling routine (AVBHOOKR) in the ACBINRTN field in the VSAM ACB. Thus, AVB receives all VSAM requests for the file until the file is closed.

ADARUN Parameters

Once AVB is active in batch, ADARUN statements in the job stream enable the job to execute VSAM application programs against Adabas databases.

You may need to modify the following ADARUN parameters in jobs using AVB. For more information about these parameters, consult the *Adabas Operations Manual.*

LU Parameter

Parameter	Specify . . .	Minimum	Maximum	Default
LU	the size of the intermediate user buffer.	4000 (see note 1)	65535	65535 (see note 2)

Notes:

1. *The minimum value may be less than 4000 if a lesser value was specified by the DBA using Adabas Online System.*
2. *The default value is set to 65535 to accommodate the length of the record buffer of the utilities that need the nucleus, e.g., ADAULD. If the value of* `LU` *is less than 65535 for an Adabas session, a response code will occur when such a utility is running.*

   The size specified must be large enough to accommodate all Adabas control information (108 bytes), all user buffers (format, record, search, value, ISN) which may be required for any given Adabas command, plus any user information passed from Adabas link routines to nucleus user exits.

   If the multifetch/prefetch option or a utility that needs large record/ISN buffers is to be used during the session, the setting of `LU` must be large enough to contain the buffers needed.

   The `LU` parameter cannot specify a value greater than the byte count implied by the `NAB` (number of attached buffers) parameter, or an error occurs. The NAB default of 16 allocates more than 64 KB.

   The prefetch mechanism in AVB allows you to specify the size of the prefetch buffer in kilobytes. If this specification increases the total size of the intermediate user buffers, a larger `LU` value may be required.

   The recommended LU setting is the present size plus the total of all PREFSZE values in the transparency table for files used in the same job step.

NISNHQ Parameter

Parameter	Specify . . .	Minimum	Maximum	Default
NISNHQ	the maximum number of records that can be placed in Hold status at the same time by one user.	1	**(NH / 4) – 1**, up to 32767	**(NH / NU) • 2** or 20, whichever is larger

NISNHQ determines the maximum number of held ISNs the hold queue (see the NH parameter description in the *Adabas Operations Manual*) can contain from one user. The most ISNs allowed on hold for a single user is 1/4 of the hold queue size, minus 1, but not more than 32,767 ISNs. The default is the greater of either the number if ISNs the hold queue can contain divided by the number of users (NU) and multiplied by two; or 20.

A user that attempts to place more records in hold status than permitted receives a non-zero response code even though there may still be space in the hold queue. This is most likely to happen when processing ranges of records. If you receive a nonzero response code related to the hold queue, increase the value of NISNHQ.

You can override this parameter setting for an individual user by specifying a different value in the Adabas control block for an OP command. See the discussion of the OP command in the *Adabas Command Reference Manual*.

OPENRQ Parameter

Parameter	Specify . . .	Possible Values	Default
OPENRQ	whether an OPEN command is required as the first command of an Adabas session.	YES | NO	YES

The default setting is not acceptable for AVB. Because AVB performs no physical OPEN in multiple-user environments or when running as an ET user, it is necessary to set OPENRQ=NO, which specifies that an OPEN command is **not** required as the first command of a session.

AVBUSER Parameters

Once AVB is active in batch, AVBUSER statements (* AVBUSER statements under VSE/ESA) and ADARUN statements in the job stream enable the job to execute VSAM application programs against Adabas databases.

AVBUSER (* AVBUSER) statements identify the transparency table to be used, the application name, and other processing characteristics. If there are no AVBUSER (* AVBUSER) statements in the job stream, all files are opened as VSAM files.

OS/390 Format

Under OS/390 or z/OS, AVBUSER statements are coded in the JCL stream either following the AVBCARDS DD statement or in a dataset referenced by the AVBCARDS DD statement.

The syntax for OS/390 or z/OS AVBUSER parameter statements is as follows:

```
AVBUSER {KEYWORD=value}  comment
```

Elements enclosed in braces { } must be concatenated; that is, no blank spaces are allowed.

A separate AVBUSER statement is required for each parameter specified.

Enter the literal AVBUSER in columns 1–7 of each parameter statement. Column 8 must be blank. Starting in column 9, enter a valid KEYWORD=value string, followed by at least one blank. There must be no blanks **within** the string. The rest of the statement may be used for comments.

Example:

The following example, which shows an AVBUSER statement with a comment, identifies the transparency table to be used in the job step:

```
AVBUSER FB=AVBTAB  TRANSPARENCY TABLE
```

VSE/ESA Format

Under VSE/ESA, the * AVBUSER statements are JCS comment cards and are presented for syntax checking and processing to the AVB job control exit that is installed when AVB is activated. The * AVBUSER statements may be included at any point before the EXEC statement.

The syntax for VSE/ESA * AVBUSER parameter statements is as follows:

```
* AVBUSER {KEYWORD[=value]}  comment
```

Elements enclosed in brackets [ ] are optional for at least one parameter. Elements enclosed in braces { } must be concatenated; that is, no blank spaces are allowed.

A separate * AVBUSER statement is required for each parameter specified.

The literal "* AVBUSER" must be entered in columns 1–9 of each parameter statement. Column 10 must be blank. Starting in column 11, there must be a valid KEYWORD or KEYWORD=VALUE followed by at least one blank. The rest of the statement may be used for comments.

Example:

The following example, which shows a * AVBUSER statement with a comment, identifies the transparency table to be used in the job step:

```
* AVBUSER FB=IVPTAB  TRANSPARENCY TABLE
```

Required AVBUSER Parameters

FB or TT

Parameter	Specify . . .	Possible Values	Default
FB or TT	the name of the Adabas transparency table to be loaded dynamically at execution time.	table-name	none

Each job step using AVB must include one and only one FB or TT parameter statement. If more than one is specified, only the last is used.

AVB scans the transparency table named in the statement and opens the corresponding Adabas files for the VSAM files listed in the table.

In a multiple-step job under VSE/ESA, each step using AVB must include an FB or TT parameter statement to reestablish the partition vector table.

LANG

Parameter	Specify . . .	Possible Values	Default
LANG	the language in which the application was written.	ASSEMBLER | COBOL | PL/I | RPGII	none

This parameter is required for RPGII applications. For languages other than RPGII, it is optional and only for documentation.

PROGRAM

Parameter	Specify . . .	Possible Values	Default
PROGRAM	the name of the application program you want to execute using AVB.	user-program-name	none

Optional AVBUSER Parameters

CLEAR or RESET (VSE/ESA Only)

Parameter	Specify . . .	Possible Values	Default
CLEAR or RESET	that the partition table is to be reset to binary zeros. This procedure is recommended for use between job steps.	none	none

Only one * AVBUSER CLEAR or * AVBUSER RESET is allowed per job step.

DBGNZ

Parameter	Specify . . .	Possible Values	Default
DBGNZ	whether AVB records all responses in the debug trace table (N) or only nonzero Adabas responses (Y).	Y | N	N

This parameter overrides the options table parameter DBGNZO for this job step.

DD

Parameter	Specify . . .	Possible Values	Default
DD	the DDNAME (DTFNAME under VSE/ESA) of a VSAM file defined in the transparency table that you want to access with VSAM for this job step instead of the corresponding Adabas file.	file-name	none

This statement overrides an AVB transparency table entry. Specify one AVBUSER DD (* AVBUSER DD) statement for each transparency table entry you want to override. Up to 20 DD statements are allowed.

For files with an AVBUSER DD (* AVBUSER DD) statement or without an entry in the transparency table, AVB passes the file-open request back to the operating system, and the files are opened as VSAM files.

ETCNT

Parameter	Specify . . .	Minimum	Maximum	Default
ETCNT	the number of Adabas update commands that can occur before AVB automatically issues an ET command.	0	999	0

If an ETCNT value greater than 999 is specified, only the first three digits are processed.

This parameter requires that MODE=ET be set. See the MODE parameter below.

If MODE=ET is specified but no ETCNT value (or 0) is specified, ETs are issued only if the application program has been modified to call AVBETBT.

Because of the differences between VSAM batch and Adabas, ETs that are generated automatically are not issued at logical syncpoints. This fact is important to remember when designing recovery/restart procedures for the application.

LOG (OS/390 or z/OS Only)

Parameter	Specify . . .	Possible Values	Default
LOG	whether AVB writes status messages to the operator console during AVB initialization.	Y | N	N

The LOG parameter is used for installation debugging.

MODE

Parameter	Specify . . .	Possible Values	Default
MODE	the Adabas user mode for the application program.	ET | EX	EX

If MODE=EX (the default), the application program functions as an exclusive user, which means that no other program or transaction can access files held by the application.

If MODE=ET, the application program functions as an ET user, which means that only specific records are held by the application; the records are released when AVB issues an ET command. If the application program has been modified to call AVBETBT, AVB issues the ET at a logical syncpoint. If not, AVB issues the ET automatically after the number of update commands specified in the ETCNT parameter; in this case, the ET is not issued at a logical syncpoint.

For information about generating ETs automatically, see page 217. For information about modifying application programs to issue an ET after the completion of a restartable unit of work, see chapter 9 starting on page 217.

TRACE

Parameter	Specify . . .	Possible Values	Default
TRACE	whether a trace will occur and if so, what type.	0–5	0

The TRACE parameter is used to turn on the batch AVB trace facility. The valid values are described in the following table:

Value	Produces . . .
0	no trace (default).
1	a one-line trace including date and time; VSAM file name; VSAM feedback code (internal); Adabas response code; and type of request.
2	a formatted dump when the VSAM feedback code equals “nn”, which is specified after TRACE=2 and a space; for example, AVBUSER TRACE=2 92 The trace 2 dump includes trace 1 output and – file information, table entry, control block (FCB), information block (FIB), and AVB buffer; – key list and password list; – transparency table entry; and – the record and the key.
3	a trace 2 output without a condition check.
4	a trace 1 output for nonzero VSAM feedback codes only.
5	a trace 3 output for nonzero VSAM feedback codes only.

No entry occurs for trace 4 or 5 if the nonzero feedback code indicates end-of-file.

AVBUSER (* AVBUSER) Statements and AVB Logic

The flow chart below, which represents AVB logic when a file is opened, illustrates the use of AVBUSER (* AVBUSER) statements:

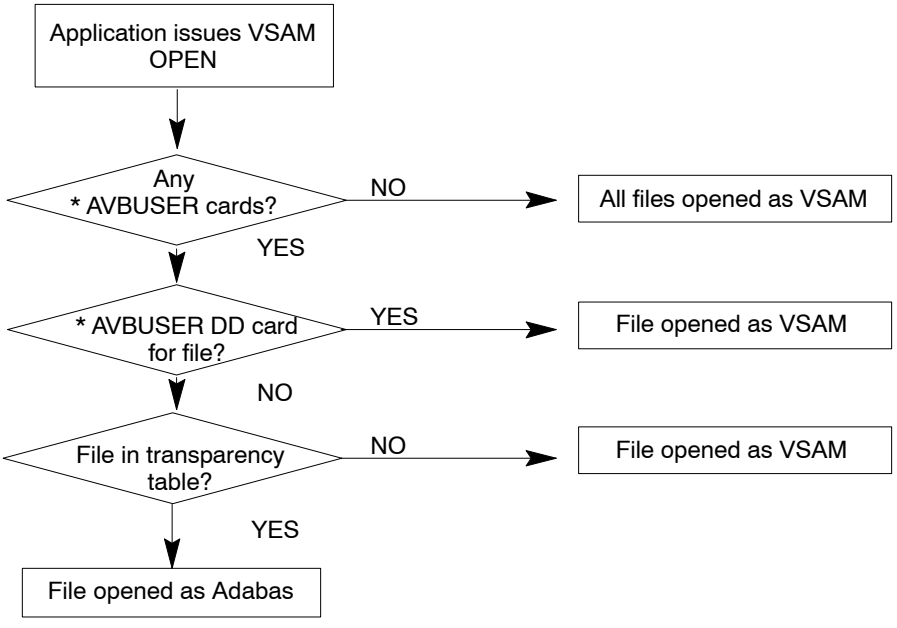


Figure 10-2: Logic for the Open Process

* *In this case, an AVBUSER FB or TT card must be present.*

OS/390 or z/OS Batch Execution JCL

Required Modifications

The following JCL changes are required in each job step that uses AVB:

- Both the Adabas load library and the AVB load library must be available to the job step. Include these libraries in the STEPLIB list, the JOBLIB list, or the system link list.
- Execute AVBLOAD to establish the AVB environment and to load and execute the user program.
- Include ADARUN parameters if you want to override the default ADARUN parameters defined for your Adabas installation.
- Include a DD statement for AVBCARDS.
- If you want AVB to print AVB statistics (AVBSTATS) and/or debug trace displays (AVBTRACE), include DD statements for these outputs.
- Include AVBUSER statements to specify the application program, transparency table, and other parameters required for your application.

Sample JCL

The following sample JCL is included as member RUNAVB in the AVB jobs library. It specifies that the user program is to run in ET-user mode and that an ET is to be issued after every 10 update commands.

To invoke AVB and execute an application program using this JCL, modify it as follows:

- In the “vrs” variables, substitute the numbers of the current version, revision, and SM level of Adabas or AVB in your batch environment.
- Add your load library to the STEPLIB list.
- In the EXEC RUNAVB statement, modify or delete the program parameter area (PARM.AVB=).
- Add the DD statements required for your application.
- Insert the name of the application program in the AVBUSER PROGRAM statement.
- If necessary, modify the name of the transparency table in the AVBUSER TT statement.
- If necessary, modify the AVBUSER MODE and AVBUSER ETCNT statements.
- Add AVBUSER DD statements for any transparency table entries you want to override.

- Supply values for your database environment in the ADARUN statements.

```
//*
//*  This sample PROC can be used as a guide for running AVB in batch.
//*
//*  The symbolic parameters used are:
//*
//*      ADALOAD   Adabas load library      (SAG.ADABAS.Vvrs.LOAD)
//*      AVBPRE    AVB HLQ prefix           (SAG.AVB)
//*      VRS       Current AVB version      (Vvrs)
//*      PRINT     SYSOUT class             ('*')
//*
//*  In addition to the above symbolic parameters, supply values for
//*  the AVBUSER and DDCARD data below. For in-stream execution
//*  place a valid JOB card at the top of this source deck.
//*  Be sure to add the loadlib containing the application program to
//*  the STEPLIB concatenation.
//*
//*****************************************************************
//RUNAVB   PROC ADALOAD='SAG.ADABAS.Vvrs.LOAD',
//              AVBPRE='SAG.AVB',
//              VRS=Vvrs,
//              PRINT='*'
//*****************************************************************
//AVB      EXEC PGM=AVBLOAD
//STEPLIB  DD DSN=&AVBPRE..&VRS..LOAD,DISP=SHR
//         DD DSN=&ADALOAD,DISP=SHR
//SYSUDUMP DD SYSOUT=&PRINT
//AVBSTATS DD SYSOUT=&PRINT                OPTIONAL: FILES STATS
//AVBTRACE DD SYSOUT=&PRINT                OPTIONAL: TRACE OUTPUT
//         PEND
//*****************************************************************
//STEPXX   EXEC RUNAVB,PARM.AVB='USER PROGRAM PARM'
//AVB.AVBCARDS DD *
AVBUSER PROGRAM=                         <=== USER PROGRAM NAME
AVBUSER LANG=COBOL
AVBUSER TT=AVBTAB                        <=== TRANSPARENCY TABLE NAME
AVBUSER MODE=ET
AVBUSER ETCNT=10
//AVB.DDCARD DD *
ADARUN DATABASE=nnn                      <=== CUSTOMIZE
ADARUN SVC=nnn
ADARUN DEVICE=nnnn
```

VSE/ESA Batch Execution JCS

To invoke the VSE batch version of AVB, modify the JCS for each job step that is to use AVB.

1. Add ADARUN cards that specify the Adabas DBID, SVC number, and device type. It is recommended that you put ADARUN control cards in a disk file to avoid problems with SYSIPT.

   The following JCS loads ADARUN cards to a DASD file for later input to a job step using AVB:

```
// ASSGN SYS004,SYSIPT
// ASSGN SYS005,DISK,VOL=USER01,SHR
// DLBL UOUT,'ADARUN.CONTROL.CARDS',0,SD
// EXTENT SYS005,USER01,1,0,0300,01
// EXEC PGM=OBJMAINT
./ CARD DLM=$$
./ COPY
ADARUN SVC=XXX
ADARUN DATABASE=YYY
ADARUN DEVICE=ZZZZ
$$
/*
```

2. Include the AVB and Adabas libraries in the LIBDEF phase search chain.

3. Include * AVBUSER statements in the job stream. The FB/TT statement is required. The following statements are optional:

   *** AVBUSER CLEAR or * AVBUSER RESET**
   *** AVBUSER DBGNZ**
   *** AVBUSER DD**
   *** AVBUSER MODE**
   *** AVBUSER ETCNT**
   *** AVBUSER TRACE**

4. Specify the VSE EXEC SIZE parameter.

   SIZE=(AUTO,200K) is recommended for most applications. You can use the following formula to calculate the GETVIS requirement in the job step. AVB is active for all partitions.

   GETVIS = Size of AVB + Size of Transparency Table + Number of Files * (4K per OPEN Control Block + 1K per CLOSE Control Block + 4K Buffer + Size of Record Area + 2K for VSAM Control Blocks + PREFETCH Size) + (MAXFILE * 160 Bytes) + (MAXTRAC * 49 Bytes.)

The present size of AVB is roughly 80K. MAXFILE and MAXTRAC are options table parameters. Add 2K for each file with alternate indices having duplicate keys.

The following JCS executes a user program with AVB:

```
// EXEC PROC=AVBvrLIB                              AVB Library
// EXEC PROC=ADAVvLIB                              Adabas Library
// LIBDEF PHASE,SEARCH=(AVBVvr.VvrLIB,ADAVv.ADALIBB),TEMP
// ASSGN SYS000,DISK,VOL=USER01,SHR
// DLBL CARD,'ADARUN.CONTROL.CARDS'                ADARUN Input Cards
// EXTENT SYS000,USER01
// ASSGN SYS007,02E                                AVB Trace Output
* AVBUSER FB=AVBTAB                               Transparency Table Name
* AVBUSER TRACE=1                                 Request Trace Output
* AVBUSER MODE=ET                                 Run in ET/BT Mode
* AVBUSER ETCNT=50                                ET After Every 50
// EXEC USERPGM,SIZE=(AUTO,200K)
```

The SYS000 and SYS007 files used in this JCS are described in the following table:

Attribute	SYS000	SYS007
DTFNAME	CARD	AVBTRAC
Function	Specify ADARUN parameters	Specify trace-output device
Device	Reader/Tape/Disk	Printer/Tape/Disk
Format		133/133/F
Required	Always; see the *Adabas Operations Manual*	When requesting a trace

To change the DTFNAME or symbolic unit name for the AVB trace-output file, use the following offsets in the AVBIOV object module, where ‘nn’ is a user-modified logical unit number and ‘xxxxxxxxxxxxxx’ is a user-modified DTFNAME specified in hexadecimal:

Verify Address	Verify Data	Replace Data	AVBRIDGE File Name
X‘0C34’	0700	nn00	SYS007
X‘0C38’	C1E5C2E3D9C1C340	xxxxxxxxxxxxxx	AVBTRAC

See job AVBZAP.X for example job control.

Running AVB and Batch Natural (OS/390 or z/OS Only)

If you want to run applications written in COBOL, PL/I, or Assembler and call AVB from batch Natural, modify the JCL as follows:

- On the EXEC JCL card, specify PGM=AVBLOAD. AVBLOAD initializes AVB.
- Specify the name of the batch Natural nucleus on the AVBUSER PROGRAM card.
- Code the necessary Natural runtime parameters on the EXEC card for AVBLOAD.
- Supply a REGION size large enough to accommodate the batch Natural nucleus, AVBLOAD, and the called application program.
- Code the Natural runtime parameter DU=ON to disable the ESTAE set by the Natural nucleus. This step prevents possible conflicts with the ESTAE set by AVBLOAD.
- Include the DDNAME for each Natural file in addition to the files needed for AVB batch operation.

The steps listed above are illustrated in the following sample JCL.

Note:
Replace "vrs" in the following JCL with the numbers of the version, revision, and SM level of Adabas, AVB, and Natural in your batch environment.

```
//        JOB
//*    This is a sample job to run an application using AVB called by a
//*    batch NATURAL program.
//*
//AVBCEXEC PROC ADALOAD='ADAvrs.LOAD',       ADABAS LOAD LIBRARY       X
//              AVBLOAD='AVBvrs.LOAD',       AVB LOADLIB               X
//              COBLIB='SYS1.VSCLLIB',       COBOL SUBROUTINE LIB      X
//              NATLIB='NATURAL.Vvrs.LOAD',  NATURAL LOAD LIBRARY      X
//              SYSOUT='SYSOUT=*'            SYSOUT CLASS
//IVP       EXEC  PGM=AVBLOAD
//STEPLIB   DD DSN=&NATLIB,DISP=SHR
//          DD DSN=&AVBLOAD,DISP=SHR
//          DD DSN=&ADALOAD,DISP=SHR
//          DD DSN=&COBLIB,DISP=SHR
//SYSMDUMP  DD &SYSOUT
//AVBSTATS  DD &SYSOUT
//AVBTRACE  DD &SYSOUT
//PRINTDD   DD &SYSOUT
//SNAPDD    DD &SYSOUT
//SYSOUT    DD &SYSOUT
```

```
//SYSPRINT  DD &SYSOUT
//SYSUT2    DD &SYSOUT,DCB=(RECFM=FA)
//CMPRINT   DD &SYSOUT
//CMPRT01   DD &SYSOUT
//CMPRT02   DD &SYSOUT
//CMSYNIN   DD DDNAME=CMSYNIN
//          PEND
//AVBCEXEC  EXEC AVBCEXEC,REGION=8192K,
//   PARM=('MT=0,DU=ON')
//IVP.AVBCARDS  DD *
AVBUSER TT=AVBTAB
AVBUSER TRACE=1
AVBUSER MODE=ET
AVBUSER ETCNT=10
AVBUSER LOG=Y
AVBUSER PROGRAM=batch-nucleus                          <=== CUSTOMIZE
/*
//IVP.DDCARD DD  *
ADARUN DA=nnn,SVC=nnn,DEVICE=device-type               <=== CUSTOMIZE
/*
//IVP.SYSUT1   DD *
      sysin data for COBOL program
/*
//IVP.CMSYNIN  DD *
LOGON userid
application-program                                    <=== CUSTOMIZE
FIN
/*
```

11

ADDING RESTART/RECOVERY SUPPORT FOR BATCH PROGRAMS

One of the advantages of ADABAS as opposed to VSAM is its support for sophisticated restart/recovery techniques at the logical transaction level. You can use ADABAS restart/recovery facilities to either

- generate ETs automatically using an AVB parameter option; or
- modify your batch programs to run as ADABAS ET logic users issuing ET/BT/RE/C1 commands at logical syncpoints.

Note:
ADABAS restart/recovery facilities do not by themselves provide the ability to restart your VSAM applications. You still must devise your own restart logic.

Generating ETs Automatically

If you do not want to modify your programs, you can generate ETs automatically by including an AVBUSER parameter in the batch JCL. In this case, the ET command is not issued at logical checkpoints (that is, after the completion of a restartable unit of work).

Issuing ETs automatically after a fixed number of ADABAS commands allows you to convert your batch programs into ET users without modifying the programs. In this case, the ET command is not issued after the completion of a restartable unit of work.

To generate ETs automatically, modify the JCL as follows:

- Add an AVBUSER (* AVBUSER for VSE/ESA) MODE=ET statement.
- Add an AVBUSER (* AVBUSER for VSE/ESA) ETCNT statement specifying the number of ADABAS update commands after which ADABAS will issue an ET.

Note:
Only commands that modify ADABAS files are counted.

Modifying Programs to Run as ET Logic Users

The most powerful way to add ADABAS restart/recovery support for VSAM batch applications is to modify the batch programs to run as ADABAS **ET logic users**.

AVB enables VSAM application programs to issue ADABAS ET, BT, and C1 commands at logical checkpoints during the execution. In addition, the RE command can be issued to read data about a completed transaction. The ability to read and write ET data in the ADABAS checkpoint file adds powerful restart/recovery capabilities to VSAM applications.

These commands are summarized below:

Cmd	Action
ET	Indicates the end of a logical transaction; commits updates to the database; stores data-protection information. Can write information about the transaction (**ET data**) to an ADABAS system file.
BT	Backs out the current transaction; removes updates made during the transaction.
RE	Reads user data stored by a previous ET command.
C1	Writes data-protection and checkpoint information to ADABAS logs.

For information about these ADABAS commands and about reading and writing ET data, consult the *ADABAS Command Reference Manual.*

Note:
For OS/390 users: *If you are converting from AVB 3.x, you must modify all application programs that call the AVBETBT module to include the new AVB–WORK parameter. This parameter is discussed in the following sections.*

OS/390

To convert a batch program into an ET logic user

1. Modify the program to call the AVBETBT entry point and pass the five positional parameter fields described in the table below.

   The third column shows the length (in bytes) and format (A=alphanumeric; B=binary; Var.=variable) of each field.

Pos	Parameter	Len/Fmt	Value
1	AVB–WORK	4A	AVB uses this reentrant work field to pass internal values.
2	FUNCTION	2A	Specify the AVB function to be performed.
3	ET–INFO		Use with the SD, RE, and ET functions:
	USER–ID	8A	Specify the ET-user ID.
	DATA–LENGTH	2B	Specify the number of bytes of ET data to be read or written.
or			
	CHECKPOINT	4A	Specify the checkpoint ID for the C1 function.
4	RESPONSE	4A	Holds the response code returned by AVB.
5	ET–AREA		Use with the SD, RE, and ET functions if ET data is being read or written:
	ET–NUMBER	4B	Holds the transaction sequence number of the last ET.
	ET–DATA	variable	Holds ET data returned by or written to ADABAS.

You do not have to use the parameter names given above, but you must pass them in the position shown. Omit the `ET–AREA` parameter if ET data is not being read or written.

2. Recompile and relink the program, including the AVBETBT module in the link step.

The following sections describe the coding of the parameters for each AVBETBT function.

VSE/ESA

To run a batch program as an ET user and issue ET, BT, RE, or C1 ADABAS commands at logical syncpoints

1. Modify the application program to call the AVBETBT entry point;
2. Recompile and relink the program, including the AVBETBT module in the link step;
3. Run the program, providing the ET data to be read or written.

Note:
Calls to AVBETBT do not by themselves provide the ability to restart. They permit you to issue the ADABAS commands ET, BT, RE, and C1 at logical points, which allows you to devise your own restart logic.

For information about ET logic and the ET, BT, RE, and C1 commands, see the *ADABAS Command Reference Manual*.

Call AVBETBT by passing four parameters. The four parameters are positional and have the following functions and formats:

Position	Purpose	Data Size	Data Type
1	Function	2 characters	Alphanumeric
2	Name or ID/Length (group) ET user ID ET data length	4 characters 8 characters 2 characters	Alphanumeric Alphanumeric Binary
3	Response	4 characters	Alphanumeric
4	ET data area	see note	see note

Note:
See the discussions of the ET, RE, and SD functions in the following sections.

Set the parameters according to the following function table:

Parameter	Values	Set By	Required
Function	ET/BT/RE/C1/SD	User	Always
Name	Any valid	User	for C1 function SYNCNAME
ET user ID	Any valid ET data user ID	User	for ET, RE, SD functions
ET data length	Length of ET data	User	for ET, RE, SD functions
Response	0XXX–100X; see note 1	AVB	Returned for each request
ET data area	see note 2	see note 2	for ET, RE, SD functions

Notes:

1. *See appendix A for information about the response codes.*
2. *See the discussions of the ET, RE, and SD functions in the following sections.*

Reading and Writing ET Data

If you want your programs to read and write ET data, call the AVBETBT entry point using the SD function **before** opening any files or issuing any ET or RE commands. You can issue batch ET, BT, and C1 commands without the SD function, but you cannot read and write ET data.

Note:
To facilitate the retrieval of ET data when the ADABAS files are opened, add the TYPE=ETOPNL entry to the batch transparency table you intend to use. In the UPD (or UPD2) parameter, add the number of each ADABAS file for which you want to read and write ET data. The ETOPNL statement and parameters are discussed on page 102.

OS/390

To perform the SD function

Call AVBETBT using the following parameters (for your convenience, lengths and formats are repeated):

Parameter	Len/Fmt	Value
AVB–WORK	4A	Internal values (set by AVB)
Function	2A	SD
ET–INFO		
USER–ID	8A	Valid ET-user ID
DATA–LENGTH	2B	Length of ET data to be read/written
Response	4A	Field not used; include for position only. Always set to character zeros on return to the application program.
ET–AREA		
ET–NUMBER	4B	Transaction sequence number of the last ET (value set by AVB).
ET–DATA	variable	ET data read from or written to ADABAS by subsequent commands. The data format is user-specified; the field length must be ≥ the length specified in the DATA–LENGTH parameter.

When you run the application program, specify the ET data to be read or written.

VSE/ESA

The SD function has the following parameter formats:

Parameter	Description
Function	SD
ID/Length:	Group-level field with two subentries:
ET user ID	– 8-byte ADABAS ET data user ID;
ET data length	– 2-byte binary ET data length to be read or written.
Response	No ADABAS call generated; character zeros or 4-byte return code if module AVBETBT detects an error.
ET data work area	Group-level field with two subentries: – 4-byte binary ET data transaction number returned on the open call to ADABAS; – ET data returned by or written to ADABAS by subsequent commands. The length must be equal to or greater than the ET data length specified above.

Issuing ET Commands

OS/390

To perform the ET function

Call AVBETBT using the following parameters (for your convenience, lengths and formats are repeated):

Parameter	Len/Fmt	Value
AVB–WORK	4A	Internal values (set by AVB)
FUNCTION	2A	ET

Parameter	Len/Fmt	Value
ET–INFO		
USER–ID	8A	Valid ET-user ID
DATA–LENGTH	2B	Length of ET data to be written
RESPONSE	4A	0xxx–100x (These response codes are described in the *ADABAS Messages and Codes* manual.)
ET–AREA		
ET–NUMBER	4B	Transaction sequence number of the last ET (value set by AVB).
ET–DATA	variable	ET data written to ADABAS by the ET command. The data format is user-specified; the field length must be greater than or equal to the length specified in DATA–LENGTH.

If you want to write ET data, the application program must request the SD function (before the files are opened) and specify the data to be written.

The ET command can be issued without writing ET data under either of two conditions:

- You do not call AVBETBT using the SD function before opening the files;
- You move binary zeros to the DATA–LENGTH field.

The skeleton COBOL program on page 230 illustrates ET calls with and without writing ET data.

VSE/ESA

The ET function issues the ADABAS ET command through the AVB batch interface. It has the following parameter formats:

Parameter	Description
Function	ET
ID/Length:	Group-level field with two subentries:
ET user ID	– 8-byte ADABAS ET data user ID;
ET data length	– 2-byte binary ET data length to be written when the ET command is processed.
Response	3-byte character representation (plus a leading zero) of the ADABAS response code returned after the execution of the ET command; or 4-byte response code returned by AVBETBT if an error is detected.
ET data work area	Group-level field with two subentries: – 4-byte binary field to hold the ET data transaction number returned after the execution of the ET command; – ET data written to ADABAS by the ET command. The length must be equal to or greater than the ET data length specified above.

The ET command may be issued without writing ET data under either of two conditions:

- AVBETBT is never called using the SD function;
- Binary zeros are moved to the ET data length field.

Issuing BT Commands

The BT function issues an ADABAS BT command to back out a transaction.

OS/390

To perform the BT function

Call AVBETBT using the following parameters (for your convenience, lengths and formats are repeated):

Parameter	Len/Fmt	Value
AVB–WORK	4A	Internal values (set by AVB)
FUNCTION	2A	BT
ET–INFO		
USER–ID	8A	Not used; include for position only.
DATA–LENGTH	2B	Not used; include for position only.
RESPONSE	4A	0xxx–100x (These response codes are described in the *ADABAS Messages and Codes* manual.)
ET–AREA		
ET–NUMBER	4B	Not used; can be omitted.
ET–DATA	variable	Not used; can be omitted.

VSE/ESA

The BT function issues the ADABAS BT command through the AVB batch interface. The parameter formats are as follows:

Parameter	Description
Function	BT
ID/Length:	Not used; supplied for position only.
Response	3-byte character representation (plus a leading zero) of the ADABAS response code returned after the execution of the BT command; or 4-byte response code returned by AVBETBT if an error is detected.
ET data work area	Not used; supplied for position only.

Issuing C1 Commands

The C1 function issues the ADABAS C1 command to perform an ADABAS checkpoint.

OS/390

To perform the C1 function

Call AVBETBT using the following parameters (for your convenience, lengths and formats are repeated):

Parameter	Len/Fmt	Value
AVB–WORK	4A	Internal values (set by AVB)
FUNCTION	2A	C1
CHECKPOINT	4A	Valid ADABAS checkpoint ID; “0000” or “SYNC” cannot be used.
RESPONSE	4A	0xxx–100x (These response codes are described in the *ADABAS Messages and Codes* manual.)

Parameter	Len/Fmt	Value
ET–AREA		
ET–NUMBER	4B	Not used; can be omitted.
ET–DATA	variable	Not used; can be omitted.

VSE/ESA

The C1 function issues the ADABAS C1 command to perform an ADABAS checkpoint. The parameter formats are as follows:

Parameter	Description
Function	C1
Name	4-byte character field containing a valid ADABAS SYNCNAME.
Response	3-byte character representation (plus a leading zero) of the ADABAS response code returned after the execution of the C1 command; or 4-byte response code returned by AVBETBT if an error is detected.
ET data work area	Not used; supplied for position only.

Issuing RE Commands

The RE function issues an ADABAS RE command to read ET data from the ADABAS checkpoint file. The RE command cannot be issued unless the SD function is issued before the files are opened.

AVB does not support the command option that allows the RE command to read ET data associated with a user ID different from the one associated with the session by the open.

OS/390

To perform the RE function

Call AVBETBT using the following parameters (for your convenience, lengths and formats are repeated):

Parameter	Len/Fmt	Value
AVB–WORK	4A	Internal values (set by AVB)
FUNCTION	2A	RE
ET–INFO		
USER–ID	8A	Valid ET-user ID
DATA–LENGTH	2B	Length of ET data to be read.
RESPONSE	4A	0xxx–100x (These response codes are described in the *ADABAS Messages and Codes* manual.)
ET–AREA		
ET–NUMBER	4B	Transaction sequence number of the last ET (value set by AVB).
ET–DATA	variable	ET data read from ADABAS by the RE command. The data format is user-specified; the field length must be greater than or equal to the length specified in DATA–LENGTH.

When you run the application program, specify the ET data to be read.

VSE/ESA

The RE function issues an ADABAS RE command through the AVB batch interface. This command allows a program to read ET data from the ADABAS checkpoint file.

Notes:

1. *The RE command cannot be issued unless the SD function is issued before the opening of the files.*
2. *AVB does not support the special command option that allows the RE command to read ET data associated with a user ID different from the one associated with the session by the open.*

The parameter formats for the RE function are as follows:

Parameter	Description
Function	RE
ID/Length:	Group-level field with two subentries:
ET user ID	– 8-byte ADABAS ET data user ID;
ET data length	– 2-byte binary field indicating the length of the ET data to be read when the RE command is processed.
Response	3-byte character representation (plus a leading zero) of the ADABAS response code returned after the execution of the RE command; or 4-byte response code returned by AVBETBT if an error is detected.
ET data work area	Group-level field with two subentries: – 4-byte binary field to pass the ET data transaction number returned after the execution of the RE command; – ET data read from ADABAS by the RE command. The length must be equal to or greater than the ET data length specified above.

Sample COBOL Program

OS/390

The following skeleton COBOL program illustrates the use of the AVBETBT entry point with the reading and writing of ET data:

```
IDENTIFICATION DIVISION.
     .
     .
  WORKING-STORAGE SECTION.
     .
     .
01   AVB-WORK                 PIC X(04)  VALUE '9999'.
01   ETBT-FUNCTION            PIC X(02)  VALUE SPACES.
01   ETBT-INFO.
     05   ET-USERID           PIC X(08)  VALUE 'USER0001'.
     05   ET-DATA-LENGTH      PIC 9(04)  COMP VALUE 10.
01   ETBT-RESP-CODE           PIC X(04)  VALUE '0000'.
01   ETBT-WORKAREA.
     05   ET-DATA-TRANSID     PIC 9(08)  COMP VALUE ZERO.
     05   ET-DATA-RECORD.
          07   ET-DATA-KEY    PIC X(10)  VALUE SPACES.
01   ETBT-SYNCNAME            PIC X(04)  VALUE 'SYN1'.
     .
     .
PROCEDURE DIVISION.
     .
     .
     MOVE 'SD' TO ETBT-FUNCTION.
     MOVE 10 TO ET-DATA-LENGTH.
     CALL 'AVBETBT' USING AVB-WORK ETBT-FUNCTION ETBT-INFO
                    ETBT-RESP-CODE ETBT-WORKAREA.
     IF ETBT-RESP-CODE NOT EQUAL '0000'
          PERFORM ETBT-ERROR-CHECK.
     .
     .
OPEN-FILES SECTION.
     .
     .
```

```
* CALL TO READ ET DATA WITH 'RE' FUNCTION.
     MOVE 'RE' TO ETBT-FUNCTION.
     MOVE 10 TO ET-DATA-LENGTH.
     CALL 'AVBETBT' USING AVB-WORK ETBT-FUNCTION ETBT-INFO
                    ETBT-RESP-CODE ETBT-WORKAREA.
     IF ETBT-RESP-CODE NOT EQUAL '0000'
          PERFORM ETBT-ERROR-CHECK.
      .
      .
* CALL TO ISSUE AN ET COMMAND AND WRITE ET DATA.
     MOVE 'ET' TO ETBT-FUNCTION.
     MOVE 10 TO ET-DATA-LENGTH.
     MOVE VSAM-KEY TO ET-DATA-KEY.
     CALL 'AVBETBT' USING AVB-WORK ETBT-FUNCTION ETBT-INFO
                    ETBT-RESP-CODE ETBT-WORKAREA.
    IF ETBT-RESP-CODE NOT EQUAL '0000'
         PERFORM ETBT-ERROR-CHECK.
      .
      .
* CALL TO ISSUE AN ET COMMAND WITHOUT WRITING ET DATA.
    MOVE 'ET' TO ETBT-FUNCTION.
    MOVE 0 TO ET-DATA-LENGTH.
    CALL 'AVBETBT' USING AVB-WORK ETBT-FUNCTION ETBT-INFO
                   ETBT-RESP-CODE ETBT-WORKAREA.
    IF ETBT-RESP-CODE NOT EQUAL '0000'
         PERFORM ETBT-ERROR-CHECK.
      .
      .
* CALL TO ISSUE A BT COMMAND  (CHECKPOINT ROLLBACK).
    MOVE 'BT' TO ETBT-FUNCTION.
    CALL 'AVBETBT' USING AVB-WORK ETBT-FUNCTION ETBT-INFO
                   ETBT-RESP-CODE ETBT-WORKAREA.
    IF ETBT-RESP-CODE NOT EQUAL '0000'
         PERFORM ETBT-ERROR-CHECK.
    .
    .
* CALL TO ISSUE C1 COMMAND (TAKE ADABAS CHECKPOINT).
    MOVE 'C1' TO ETBT-FUNCTION.
    MOVE 'SYN1' TO ETBT-SYNCNAME.
    CALL 'AVBETBT' USING AVB-WORK ETBT-FUNCTION ETBT-SYNCNAME
                   ETBT-RESP-CODE ETBT-WORKAREA.
    IF ETBT-RESP-CODE NOT EQUAL '0000'
         PERFORM ETBT-ERROR-CHECK.
    .
    .
```

```
ETBT-ERROR-CHECK.
    DISPLAY 'AVBETBT RESP: ' ETBT-RESP-CODE.
    IF ETBT-RESP-CODE LESS THAN '1000'
         GO TO CHECK-ADABAS-RESP.
    IF ETBT-RESP-CODE EQUAL '1001'
         DISPLAY 'AVBRIDGE MODULE NOT PROPERLY LOADED.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1002'
         DISPLAY 'AVBUSER MODE=ET CARD NOT GIVEN.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1003'
         DISPLAY 'INVALID AVBETBT FUNCTION SPECIFIED.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1004'     (*** VSE users only)
         DISPLAY 'CDLOAD FOR AVBPVT MODULE FAILED.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1005'
         DISPLAY 'OPTIONS TABLE AVBOPT NOT LOADED.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1006'
         DISPLAY 'INVALID ETBT-INFO PARM GIVEN.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1007'
         DISPLAY 'ET DATA LENGTH INVALID.'
           GO TO ERR-EXIT.
    IF ETBT-RESP-CODE EQUAL '1008'
         DISPLAY 'INVALID ETBT-WORKAREA PARM GIVEN.'
           GO TO ERR-EXIT.
    .
    .
CHECK-ADABAS-RESP.
    .
    .
ERR-EXIT.
    .
    .
    GOBACK.
```

VSE/ESA

The following skeleton COBOL program illustrates the use of the AVBETBT entry point with the reading and writing of ET data:

```
IDENTIFICATION DIVISION.
     .
     .
WORKING—STORAGE SECTION.
     .
     .
01   ETBT—FUNCTION            PIC X(02)  VALUE SPACES.
01   ETBT—INFO.
     05   ET—USERID           PIC X(08)  VALUE 'USER0001'.
     05   ET—DATA—LENGTH      PIC 9(04)  COMP VALUE 10.
01   ETBT—RESP—CODE           PIC X(04)  VALUE '0000'.
01   ETBT—WORKAREA.
     05   ET—DATA—TRANSID     PIC 9(08)  COMP VALUE ZERO.
     05   ET—DATA—RECORD.
          07   ET—DATA—KEY    PIC X(10)  VALUE SPACES.
01   ETBT—SYNCNAME            PIC X(04)  VALUE 'SYN1'.
     .
     .
PROCEDURE DIVISION.
     .
     .
     MOVE 'SD' TO ETBT—FUNCTION.
     MOVE 10 TO ET—DATA—LENGTH.
     CALL 'AVBETBT' USING ETBT—FUNCTION ETBT—INFO
                    ETBT—RESP—CODE ETBT—WORKAREA.
     IF ETBT—RESP—CODE NOT EQUAL '0000'
          PERFORM ETBT—ERROR—CHECK.
     .
     .
OPEN—FILES SECTION.
     .
     .
* CALL TO READ ET DATA WITH 'RE' FUNCTION.
     MOVE 'RE' TO ETBT—FUNCTION.
     MOVE 10 TO ET—DATA—LENGTH.
     CALL 'AVBETBT' USING ETBT—FUNCTION ETBT—INFO
                    ETBT—RESP—CODE ETBT—WORKAREA.
     IF ETBT—RESP—CODE NOT EQUAL '0000'
          PERFORM ETBT—ERROR—CHECK.
     .
     .
```

```
* CALL TO ISSUE AN ET COMMAND AND WRITE ET DATA.
     MOVE 'ET' TO ETBT-FUNCTION.
     MOVE 10 TO ET-DATA-LENGTH.
     MOVE VSAM-KEY TO ET-DATA-KEY.
     CALL 'AVBETBT' USING ETBT-FUNCTION ETBT-INFO
                    ETBT-RESP-CODE ETBT-WORKAREA.
     IF ETBT-RESP-CODE NOT EQUAL '0000'
          PERFORM ETBT-ERROR-CHECK.
     .
     .
* CALL TO ISSUE AN ET COMMAND WITHOUT WRITING ET DATA.
     MOVE 'ET' TO ETBT-FUNCTION.
     MOVE 0 TO ET-DATA-LENGTH.
     CALL 'AVBETBT' USING ETBT-FUNCTION ETBT-INFO
                    ETBT-RESP-CODE ETBT-WORKAREA.
     IF ETBT-RESP-CODE NOT EQUAL '0000'
          PERFORM ETBT-ERROR-CHECK.
     .
     .
* CALL TO ISSUE A BT COMMAND  (CHECKPOINT ROLLBACK).
     MOVE 'BT' TO ETBT-FUNCTION.
     CALL 'AVBETBT' USING ETBT-FUNCTION ETBT-INFO
                     ETBT-RESP-CODE ETBT-WORKAREA.
     IF ETBT-RESP-CODE NOT EQUAL '0000'
          PERFORM ETBT-ERROR-CHECK.
     .
     .
* CALL TO ISSUE C1 COMMAND (TAKE ADABAS CHECKPOINT).
     MOVE 'C1' TO ETBT-FUNCTION.
     MOVE 'SYN1' TO ETBT-SYNCNAME.
     CALL 'AVBETBT' USING ETBT-FUNCTION ETBT-SYNCNAME
                    ETBT-RESP-CODE ETBT-WORKAREA.
     IF ETBT-RESP-CODE NOT EQUAL '0000'
          PERFORM ETBT-ERROR-CHECK.
     .
     .
 ETBT-ERROR-CHECK.
     DISPLAY 'AVBETBT RESP: ' ETBT-RESP-CODE.
     IF ETBT-RESP-CODE LESS THAN '1000'
          GO TO CHECK-ADABAS-RESP.
     IF ETBT-RESP-CODE EQUAL '1001'
          DISPLAY 'AVBRIDGE MODULE NOT PROPERLY LOADED.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1002'
          DISPLAY '* AVBUSER MODE=ET CARD NOT GIVEN.'
            GO TO ERR-EXIT.
```

```
     IF ETBT-RESP-CODE EQUAL '1003'
          DISPLAY 'INVALID AVBETBT FUNCTION SPECIFIED.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1004'     (*** VSE users only)
          DISPLAY 'CDLOAD FOR AVBPVT MODULE FAILED.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1005'
           DISPLAY 'OPTIONS TABLE AVBOPT NOT LOADED.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1006'
          DISPLAY 'INVALID ETBT-INFO PARM GIVEN.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1007'
          DISPLAY 'ET DATA LENGTH INVALID.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1008'
          DISPLAY 'INVALID ETBT-WORKAREA PARM GIVEN.'
            GO TO ERR-EXIT.
     IF ETBT-RESP-CODE EQUAL '1009'
          DISPLAY 'AVB ENVIRONMENT NOT CORRECTLY ESTABLISHED.'
            GO TO ERR-EXIT.
     .
     .
 CHECK-ADABAS-RESP.
     .
     .
 ERR-EXIT.
     .
     .
     GOBACK.
```

Note:
A dynamic call must be used to call AVBETBT when using a COBOL program linked RMODE=ANY. This allows the AVBETBT phase to be loaded below the 16MB line. For example:

```
.
.
01   PROGRAM-ETBT  PIC X(08)
.
.
.
MOVE 'AVBETBT' TO PROGRAM-ETBT.
CALL PROGRAM-ETBT USING . . . .
```

Ensuring Data Integrity in Batch ET Mode

When you run a batch program in ET mode, the files it uses can be updated concurrently by the batch program and by an online transaction. Although ADABAS does not fulfill an update request for a record if another application is holding the record for update, data integrity can be compromised when files are opened concurrently for update by more than one application.

Therefore, when updating files from batch and online concurrently, it is recommended that you take enough checkpoints to enable you to recover the files and restart the process after an abnormal ending.

Note:
AVB makes it possible to run a batch application in ET mode; it is the user's responsibility to devise the procedure to ensure data integrity and recovery/restart when required.

Special Considerations

- **Processing Files Already Converted to ADABAS**

  Comment out the DLBL and EXTENT statements for files already converted to ADABAS files.

- **Ignoring Permanent Assignments for SYS007**

  If the partition in which a program using AVB is to run has permanent assignments for SYS007, include the following JCS card to ignore the permanent assignments:

  ```
  // ASSGN SYS007,IGN
  ```

- **Specifying a Transparency Table for Each Job Step**

  In a multiple-step job, a * AVBUSER FB=xxxxxx statement (where xxxxxx is the name of the transparency table) must be specified for each step that uses AVB.

- **Using FAQS with AVB**

  If the product FAQS (Goal Systems) is in use, remove the VSAM OPEN and CLOSE transients ($$BOVSAM and $$BCVSAM) from the FAQS directory (FTL). SOFTWARE AG recommends that you activate AVB before you activate FAQS.

- **Using VSAM TUNE with AVB**

  If the product VSAM TUNE (MACRO IV) is in use, activate AVB before VSAM TUNE.

- **Handling SYSIPT File Sequence**

  To avoid problems created by the presence of multiple SYSIPT files, keep ADARUN cards in a disk file.

12

DEBUGGING AVB APPLICATIONS

This chapter discusses general debugging for AVB under OS/390 and VSE/ESA including

- using the AVTR transaction or AVBTRACE to debug AVB.
- creating and interpreting an IPCS-formatted dump dataset under OS/390; and
- using CEDF to debug AVB version 5.1 under CICS/TS.

First Steps

Table Definitions

Most application ABENDs are caused by incorrect or incomplete system or file table definitions. Incorrect or missing entries in the options or transparency tables cause unpredictable application behavior and usually an ABEND. Always verify these entries before proceeding with application or AVB debugging.

Debug Trace Table and Adabas Control Blocks

The AVB debug trace table and Adabas control blocks help in debugging an application ABEND by displaying the most recent Adabas command and its results.

The trace table is a wraparound table of Adabas commands and response codes. Its size is determined by the MAXTRAC entry in the options table.

The table always contains the most current trace entries: one entry per Adabas request. It is identified in a dump by the header “DEBUG TABLE BEGIN” followed by the table entries.

AVB marks the next available entry in the table with 16 Ns (that is, “NNNNNNNNNNNNNNNN”). The entry just before this mark is the command most recently completed by Adabas.

Viewing the Trace Table Online or in Batch

Sometimes an AVB problem occurs because a program using AVB does not interpret errors in the Adabas processing correctly.

You can use the AVTR transaction under CICS (see page 246) and the AVBTRACE dataset in AVB batch sessions to determine whether any meaningful nonzero Adabas response codes have been recorded in the trace table. In most cases, correcting the reported Adabas problem will also correct the AVB behavior.

Debugging Under OS/390 or z/OS

To debug AVB version 5.1 applications under OS/390 or z/OS, you need a dump that includes the global areas of memory (CSA, SQA, link pack areas, etc.) where AVB stores some of its runtime information. Software AG recommends IBM's Interactive Problem Control System (IPCS) for analyzing AVB dumps.

Note:
*See also the section **Debugging under CICS/TS** on page 245.*

Creating Dump Datasets for Use with IPCS

When an error occurs in the CICS address space, CICS/TS normally writes an SVC dump. This dump contains the required information and can be processed by IPCS.

In batch, the JCL statement SYSMDUMP DD writes an SVC-type dump to a specified dataset. In addition to the private area where the error took place, this dump contains the required global areas. It can be processed by IPCS.

IPCS **cannot** process dumps created by the SYSABEND DD and SYSUDUMP DD batch JCL statements. In addition, these dumps do not contain the required global areas.

Important:
*If you want your Software AG technical support representative to analyze an AVB version 5.1 dump, you must provide a tape with an SVC dump (under CICS/TS) or SYSMDUMP (in batch). Only dumps that include **at least** dump options CSA, LPA, and NUC can be accepted.*

Using GDG to Manage Multiple Dump Datasets

Certain error scenarios create multiple dumps. This is the case when an error is not serious enough to halt processing but corrupts the information needed for further processing. Another dump occurs, possibly spawning further dumps.

The JCL disposition DISP=MOD, which causes subsequent dumps to be appended to the previous ones, is incompatible with a dataset written by SYSMDUMP. Thus, multiple dump datasets must be managed as part of a Generation Data Group (GDG). A GDG is a collection of physical sequential datasets that are cataloged as a group under the same name.

Instead of the member names used in PDS processing, GDG members are referred to by their relation to the current generation. For example, the current generation is designated (+0), and the generation just preceding it is (–1). A new generation is (+1).

You can use the member DEFGDG in the AVB 5.1 jobs library to define a Generation Data Group for your IPCS dumps. The member USEGDG provides a sample for creating the SYSMDUMP statement in your task's JCL.

For more information about GDGs and the parameters used to define one in the DEFGDG member, consult the MVS/DFP or DFSMS/MVS manual *Using Data Sets*.

Using IPCS to Interpret an AVB Version 5.1 Dump

1. Set up the IPCS environment.

   IPCS runs in a TSO/ISPF session. Normally, a site has a CLIST that users run to set up the IPCS environment. An example of a CLIST follows:

   ```
   CONTROL NOMSG FREE F(IPCSPARM IPCSPRNT) EX
   'SYS1.SBLSCLI0(BLSCLIBD)' FREE F(IPCSPARM IPCSPRNT)
   ```

2. Allocate an IPCS dump directory before starting IPCS.

   IPCS uses the dump directory to track the status of the dumps currently being investigated. See IBM's *IPCS Command Reference* manual for information about creating a dump directory if one is not already allocated.

3. Start IPCS.

4. On the IPCS Main Menu, select option 0 to verify the name of the dump dataset before processing.

   The dump dataset name is included in the system messages output of the ABENDing task.

5. Modify the dataset name as necessary; then press ENTER to update the dump directory.

6. Select option 6 from the Main Menu to display the Commands Menu, which lists commands that can be used to analyze the data in the dump.

7. Issue the command STATUS to display a summary of the problem.

   The summary provides information about the program status word (PSW) and register values at the time of the ABEND. Record this information for your Software AG technical support representative.

   Because the STATUS information is frequently inconclusive, especially under CICS/TS, the following sections describe other IPCS commands that are useful when debugging AVB in a CICS/TS or batch environment.

IPCS Commands for Both CICS/TS and Batch Environments

The following IPCS commands will regularly be used for both CICS and batch environments:

Option	Use this command to . . .
LPAMAP	list the names and addresses of all modules in the link pack area (LPA). AVB 5.1 does not currently load any modules into the LPA or common storage.
COPYDUMP	determine how many dumps have actually been written to SYSMDUMP. You can also use it to copy the multiple dumps into separate datasets. This condition is most likely to exist when the SYSMDUMP has been coded for a CICS region.
FIND	search for a string or address in the dump.
LIST	display memory locations in the dump. The BROWSE option, which is discussed in the section **Using IPCS to Browse the Dump** on page 244, is a more convenient way to display memory.

VERBEXIT Command for a CICS/TS Environment

CICS/TS supplies special VERBEXIT commands for each CICS release. These VERBEXIT commands format CICS-specific areas in the dump.

The basic command is VERBEXIT DFHPDvr0 where “vr” is the version and release level of the CICS/TS you are using. This will probably give you too much information. The following parameters can be used to limit the information:

Option	Trace Information Displayed
TR	All CICS trace information.
FCP	File control information, such as the access-method control block (ACB) and request parameter list (RPL).
UEH	Information about CICS exits (AVB version 5.1 has AVBCICSA, AVBCICS2, AVBCICS3, and AVBREQT exits).
KE	Information about the CICS kernel, where program status word (PSW) and register values normally are found.
XM	Details about the EXEC interface components.

Enclose the parameters in apostrophes suffixed to the command:

```
VERBEXIT DFHPDvr0 'UEH' 'KE'
```

Useful Commands for a Batch Environment

Since IPCS was originally designed for non-CICS applications, most of the commands (other than VERBEXIT) displayed on the Commands Menu can be helpful in debugging a normal SYSMDUMP. The following commands are particularly useful:

Option	Trace Information Displayed
STATUS	PSW and register information.
SUMMARY	Summary information about task control blocks (TCBs) and request blocks (RBs) involved in the current error.
LISTMAP	Information about all known structures (control blocks, modules, etc.), including the address in the dump.

Using IPCS to Browse the Dump

The BROWSE option on the IPCS Main Menu provides the easiest way to view the raw data in the dump. The Browse panel allows you to specify a starting address for your browse session. You can obtain such addresses from the LPAMAP, LISTMAP, or VERBEXIT DFHPDvr0 output.

After your browse session has begun, use the LIST subcommand to change the starting address. The PF7 and PF8 keys page forward and backward in the data. Other useful PF key commands are listed at the bottom of the screen.

You can advance through the dump by specifying an offset rather than an address on the LIST subcommand. Designate an offset by prefixing a plus sign to the offset value.

Note:
*By default, **all** addresses and offsets are specified in hexadecimal notation.*

Frequently, the data you view in a dump is a control block, which may contain addresses of other areas or other control blocks. To advance to an address displayed on the screen, use the TAB key to position the cursor directly in front of the displayed address. Type a percent sign (%) if the address is below the 16-MB line or a question mark (?) if the address is above the line (if the address is above the line, the high-order byte contains a nonzero number):

```
.
.
00B57234              00280001   00000000   00000000   | .   ............ |
00B57240   00000000   C3C9C3E2   D9C44040   005952E9   | ....CICSRD  ...Z |
00B57250 % 0015E080   00000000   00000000   00000000   | ..\............. |
.
.
```

When you press ENTER, the screen displays the data at the address marked by the percent sign:

```
0015E080   C1E5C2C3   C9C3E2E2   E5F44BF1   4BF1D6E2   | AVBCICSSV4.2.1OS |
0015E090   40F0F761   F1F461F9   F4010000   00000000   |  05/14/96....... |
0015E0A0   00000000   8005EBA0   07558000   00164070   | ............. . |
.
.
```

You can use the FIND subcommand to locate either character strings or addresses in the dump.

Debugging Under VSE/ESA

When seeking debugging assistance from your Software AG technical support representative, you need to provide

- the console log showing AVB activation messages;
- the console log for the failing job or jobstep; and
- any storage dumps taken for the failure.

In addition, it may be necessary to provide a listing of the phases in the SDL, particularly the AVB $$B transients and the AVB partition vector table.

When debugging nonzero Adabas response codes, nonzero VSAM feedback codes, or incorrect data output, the AVB trace facility is invaluable. See the section **Viewing the Trace Table Online or in Batch** on page 240.

The debugging and information gathering techniques for VSE CICS/TS are generally the same as those discussed in the section **Debugging under CICS/TS** on page 245.

General Debugging Under CICS/TS

The following sections describe general debugging and information gathering techniques for CICS/TS. They may be used on OS/390 (z/OS) or VSE/ESA with AVB version 5.1.

Using CEDF to Debug AVB 5.1 Under CICS/TS

When a program error occurs, intermediate output (such as console messages or screen updates) may be insufficient to indicate how much processing actually occurred before the error. You can use the CICS Execution Diagnostic Facility (CEDF) to verify some AVB processing.

Note:
*The **CEDF** command may be protected by security at your site.*

Every time AVB performs an action against an Adabas file, the task-related user exit (TRUE) of the Adabas command-level link routine issues an EXEC CICS WAIT EXTERNAL.

Before issuing the failing CICS/AVB transaction

☐ Issue the command CEDF.

In addition to your transaction program's normal output, you will see "before" and "after" snapshots of the CICS EXEC interface processing for each command-level request issued by your transaction. Wherever the transaction accesses a bridged VSAM file, you should see the EXEC CICS WAIT EXTERNAL (representing the call to Adabas) from the task-related user exit.

Using AVTR to Debug AVB

To display AVB trace information

☐ Execute the AVTR transaction.

```
                Adabas Bridge for VSAM Trace Facility
            AVBridge V5.1.1                    2000—11—23  16:21:46

IBL  RBL  Function Cmd Option  File     FNR   VSB RSP KL    ISN   TFCTR  P Term
0000 0088 FIND     L9  EQGT    SYS021   01002 00  000 007 00000000  0C     0031
0000 0080 GET NE   RC          SYS021   01002     000 000 00000000  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000808  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000809  0E     0031
0000 0080 FIND     RC  EQGT    SYS021   01002     000 007 00000000  14     0031
0000 0088 FIND     L9  EQGT    SYS021   01002 00  000 007 00000000  14     0031
0000 0080 GET NE   RC          SYS021   01002     000 000 00000000  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000824  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000825  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000826  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000827  0E     0031
0000 0080 GET NE   L3          SYS021   01002 00  000 000 00000828  0E     0031
0000 0080 GET NE   L3          SYS021   01002 10  003 000 00000828  0E     0031
0000 0080 ET/RC    ET                   01002     000 000 00000000  12     0031
0000 0080 ET/RC    RC                   01002 00  000 000 00000000  12     0031


 PF2=Refresh PF3=Quit PF4=Print PF5=Fixes PF6=Bottom PF7=Backward PF8=Forward
```

The columns of the trace table display the following information for each Adabas request:

Column	Description
IBL	ISN buffer length
RBL	record buffer length
Function	internal AVB function code
Cmd	generated Adabas command
Option	internal AVB option code
File	file name
FNR	Adabas file number
VSB	VSAM status code
RSP	Adabas response code
KL	key length
ISN	ISN value
TCAFCTR	CICS request code
P	Command option 1 when PREFETCH is active
Term	terminal ID originating the task (or the characters “ATPI” when the transaction was not initiated from a terminal)

To display the list of fixes applied to AVB

☐ Press PF5 from the trace table display screen.

AVB Online Services provides a more detailed analysis of AVB trace information. See the chapter **AVB Online Services** starting on page 163 for more information.

Interpreting Error Messages

When interpreting an error message, consider the context in which it appears. You may need to explore the context to find the underlying cause of an AVB problem.

For example, during CICS startup, the following message might appear in the system messages:

DFHFC0953 yourcics OPEN or CLOSE of file AVBFILE failed. CICS logic error 8798,2AD4

You know that AVBFILE is a bridged file because it is in your AVB transparency table. Because CICS has failed to start, AVB appears to be the problem. However, Adabas has returned a response code 17 (invalid file number) or 148 (inaccessible or inactive database) for the bridged Adabas file. Both of these response codes are caused by the user or the environment rather than by AVB.

The CICS message correctly stated the problem, but the real question is, “What would prevent an Adabas file from opening correctly?” The database may not be active, or the network path to the database may be down. The wrong DBID or FNR may have been specified in the transparency-table entry for AVBFILE. CICS does not start if a file that must be opened during startup is missing.

13

MESSAGES AND CODES

This chapter contains AVB messages, ABEND codes, and ET user response codes and tells you how to interpret the messages issued by AVB Online Services.

For Adabas LNKENAB (ADAKnnn) messages, see the *Adabas Messages and Codes* manual.

AVBAnnn – VSE/ESA Batch Activation Messages

These messages are written to the operator console (if any error is detected when AVB is activated) and to SYSLST.

AVBA001 ADABAS BRIDGE FOR VSAM V5.1.n IS ACTIVE

Explanation: The normal execution of program AVBSWI.

System Action: AVB is ready to convert VSAM requests into Adabas calls as requested in AVBUSER control statements.

User Action: None.

AVBA002 AVBPVT PHASE NOT FOUND

Explanation: Phase AVBPVT was not found.

System Action: AVB cannot be activated.

User Action: Verify that phase AVBPVT is present in a concatenated library (see also AVBA003).

AVBA003 AVBPVT PHASE IS NOT IN THE SVA

Explanation: AVBPVT was not loaded in the SVA.

System Action: AVB cannot be activated.

User Action: Verify that phase AVBPVT was loaded in the SVA (using the SET SDL command). Load the phase if required and execute phase AVBSWI.

AVBA004 PAGE FIX IN SVA FAILED FOR AVBPVT

Explanation: The attempt to page fix AVBPVT failed.

System Action: AVB cannot be activated.

User Action: Contact your Software AG technical support representative.

AVBA005 UNABLE TO LOCATE $$BOVSAM IN SDL

Explanation: An entry for $$BOVSAM (the B-transient for a VSAM open) was not found in the SDL.

System Action: AVB cannot be activated.

User Action: Verify that $$BOVSAM has an entry in the SDL.

AVBA006 UNABLE TO LOCATE $$BCVSAM IN SDL

Explanation: An entry for $$BCVSAM (the B-transient for a VSAM close) was not found in the SDL.

System Action: AVB cannot be activated.

User Action: Verify that $$BCVSAM has an entry in the SDL.

AVBA007 UNABLE TO LOCATE $$BAVBO1 IN SDL

Explanation: An entry for $$BAVBO1 (the B-transient for an AVB open) was not found in the SDL.

System Action: AVB cannot be activated.

User Action: Verify that $$BAVBO1 has an entry in the SDL. (See the chapter **Batch VSE/ESA Installation**.)

AVBA008 UNABLE TO LOCATE $$BAVBC1 IN SDL

Explanation: An entry for $$BAVBC1 (the B-transient for an AVB close) was not found in the SDL.

System Action: AVB cannot be activated.

User Action: Verify that $$BAVBC1 has an entry in the SDL. (See the chapter **VSE/ ESA Installation**.)

AVBA009 UNABLE TO LOCATE VSAM TRANSIENTS IN SDL

Explanation: The pointers to the VSAM and AVB transients in the SDL were corrupted.

System Action: AVB cannot be activated/deactivated.

User Action: Check whether any other product is interacting with VSAM at open or close time. Perform an IPL and contact your Software AG technical support representative if the problem persists.

AVBA010 UNABLE TO DETERMINE VSE LEVEL

Explanation: Information about VSE/ESA (version and release) could not be found in the supervisor.

System Action: AVB cannot be activated. A core dump is produced.

User Action: Obtain VSE version and release level; contact your Software AG technical support representative.

AVBA011 ADABAS BRIDGE FOR VSAM DEACTIVATED

Explanation: AVB was active but was deactivated by the execution of program AVBSWI.

System Action: AVB does not convert any VSAM request if it is not reactivated by executing AVBSWI.

User Action: None.

AVBCnnn – OS/390 Messages at Batch Close Time

AVBC098 CLOSE ERROR ADABAS RSP XXX STATUS YY

Explanation: It was not possible to set an equivalent status for the Adabas response code. The VSAM status is set to 92 and the Adabas RSP indicates the Adabas error.

AVBC099 INTERNAL ERROR

Explanation: An illogical condition occurred. In the same line of the message one of the following causes is indicated:

ACB WAS NOT OPENED

INVALID ACCESS MODE IN ACB

System Action: An ABEND and a full partition dump are forced.

AVBEnnn – Messages from Batch AVB Job Control Exit

These messages are written on the same line as the AVB control cards starting in column 29.

AVBE020 AVBPVT NOT FOUND IN THE SDL

Explanation: An entry for phase AVBPVT was not found in the SDL.

System Action: The AVBUSER control cards cannot be processed.

User Action: Verify that phase AVBPVT was loaded in the SVA (using the SET SDL command). Load the phase if required, activate AVB, and resubmit the job.

AVBE021 INVALID AVBUSER CONTROL STATEMENT

Explanation: A syntax error occurred.

System Action: The statement is not processed.

User Action: Verify the syntax and resubmit the job.

AVBE022 AVBUSER MODE PARAMETER INVALID

Explanation: A syntax error occurred in the parameter specifying the run mode (MODE=ET or MODE=EX are valid).

System Action: The statement is not processed.

User Action: Verify the syntax and resubmit the job.

AVBE023 NO ROOM IN DD NAME TABLE

Explanation: The number of AVBUSER statements that override entries in the transparency table and access VSAM files instead of Adabas files has exceeded the maximum allowed (i.e., 20 AVBUSER DD statements).

System Action: The statements in error are not processed.

User Action: Do not use more than 20 AVBUSER DD statements.

AVBE025 INVALID COUNT PARM FOR ET MODE

Explanation: The AVBUSER ETCNT control statement has a syntax error.

System Action: The `ETCNT` value is set to 0, and no ET commands are being issued automatically. Only requests for ET using calls to AVBETBT are accepted.

User Action: Correct the AVBUSER ETCNT parameter statement and resubmit the job.

AVBInnn Messages

VSE Batch Initialization Messages

These messages are written to the operator console and to SYSLST.

AVBI001 ADABAS BRIDGE FOR VSAM Vv.r.s INSTALLED

Explanation: The AVB job control exit is active.

System Action: AVBUSER control cards are processed. However, AVB itself is not active.

User Action: None.

AVBI091 UNABLE TO LOCATE $JOBEXIT IN SDL

Explanation: The entry for phase $JOBEXIT was not found in the SDL.

System Action: AVB cannot be installed.

User Action: Get information on the version/level/system modification of the operating system and contact your Software AG technical support representative.

AVBI092 UNABLE TO LOCATE AVBJBXT IN SDL

Explanation: The entry for phase AVBJBXT (job control exit) was not found in the SDL.

System Action: AVB cannot be installed.

User Action: Verify that phase AVBJBXT was loaded in the SVA.

AVBI093 PHASE AVBJBXT IN ERROR

Explanation: The phase AVBJBXT loaded in the SVA is not the job control exit supplied with AVB.

System Action: AVB cannot be installed.

User Action: Verify a probable error during link for AVBJBXT.

AVBI094 ADABAS BRIDGE FOR VSAM ALREADY INSTALLED

Explanation: The program AVBINST, used to install the AVB job control exit, can be executed only once.

System Action: AVB cannot be installed.

User Action: None if AVB was actually installed. If not, perform an IPL and reestablish the job control exit chain.

AVBI095 UNABLE TO DETERMINE VSE LEVEL

Explanation: Information about VSE/ESA (i.e., version and release) could not be found in the supervisor.

System Action: AVB cannot be initialized. A core dump is produced.

User Action: Obtain the VSE version and release level; contact your Software AG technical support representative.

AVBI100 GETVIS FAILED FOR {FILE | EXUFL | TRACE | DYNAREA} TABLE

Explanation: A GETVIS instruction was unable to acquire storage.

System Action: AVB is terminated.

User Action: Increase the partition size, adjust the `SIZE` parameter on the EXEC card to allocate more GETVIS storage, or reduce the `MAXTRAC` option in the options table (AVBOPT). Contact your Software AG technical support representative if necessary.

AVBLnnn – OS/390 Batch Activation Messages

These messages are written to the operator console and to the programmer message set if an error occurs during batch activation.

AVBL090 ERROR LOADING OPTIONS TABLE RC=xx

Explanation: An error occurred during the loading of module AVBOPT; “xx” is the return code in register 15.

User Action: Verify that the options table has been successfully linked and that its library is accessible.

AVBL100 ERROR LOADING AVBRIDGE RC=xx

Explanation: An error occurred during the loading of module AVBRIDGE; “xx” is the return code in register 15.

User Action: Verify that the AVB library is accessible.

AVBL200 ERROR EXECUTING ESTAE RC=xx

Explanation: An ESTAE macro was issued; “xx” is the error return code.

User Action: Contact your Software AG technical support representative.

AVBL300 ERROR LOADING TRANSPARENCY TABLE RC=xx

Explanation: The module name specified on the AVBUSER FB or TT parameter card either does not exist or is not on a library in the current load list.

AVBL400 GETMAIN FAILED FOR WORK AREAS RC=xx

Explanation: A GETMAIN macro failed to acquire storage for file work areas, for the file table (`MAXFILE`) or the trace table (`MAXTRAC`).

User Action: Increase the region size or reduce the `MAXFILE` or `MAXTRAC` entries requested in the options table.

AVBL500 GETMAIN FAILED FOR MAIN TABLE RC=xx

Explanation: A GETMAIN macro failed to acquire storage for the SVC screening table; *xx* is the return code in register 15. The GETMAIN is attempting to acquire 2048 bytes from subpool 253.

User Action: Contact your system programmer or your Software AG technical support representative for assistance.

AVBL700 ERROR LOADING USER PROGRAM RC=xx

Explanation: The program specified on the `AVBUSER PROGRAM` statement does not exist, or its load library is not accessible.

AVBL800 NO USER PROGRAM SPECIFIED

Explanation: No program name was specified on the `PROGRAM` parameter card.

AVBL900 INVALID TRANSPARENCY TABLE LOADED

Explanation: The module named in the `AVBUSER FB` or `TT` statement is not a valid AVB transparency table.

AVBOnnn – Messages at Batch Open Time

These messages are written to the operator console.

AVBO001 AVBRIDGE INIT ERROR reason

In the same line of the message one of the following causes is indicated:

reason GETVIS

Explanation: The open routine was unable to acquire GETVIS storage for file work areas.

User Action: Increase the partition GETVIS area or reduce the file and/or trace table size (MAXTRAC and MAXFILE in AVBOPT).

reason CDLOAD

Explanation: The CDLOAD for Adabas VSAM code was not successful. An attempt to load one of the following phases was rejected: AVBRIDGE, AVBTAB, AVBOPT.

User Action: See the IBM manual *VSE Macro User's Guide* for possible causes and contact you Software AG technical support representative.

reason AVBTAB – INVALID TRANSPARENCY TABLE

Explanation: The phase, cataloged under the name of the transparency table to use for this run, does not have the identifier for a valid transparency table.

reason DYNAREA

Explanation: Internal error during dynamic area chaining.

User Action: Contact your Software AG technical support representative.

reason RESET

Explanation: In a multiple-step job, a previous step used the AVB series, but no `*AVBUSER RESET` or `*AVBUSER FB=xxxxxxxx` was specified for this step.

System Action: At the first open request for a VSAM file, an implicit RESET is performed. (xxxxxxxx is the name of the transparency table.)

User Action: Specify the * AVBUSER card for this step.

AVBO098 OPEN ERROR ADABAS RSP XXX VSAM STATUS YY

Explanation: It was not possible to set an equivalent VSAM status for the Adabas response code. The VSAM status is set to 92 and the Adabas RSP indicates the actual Adabas error.

AVBO099 INTERNAL ERROR

Explanation: An illogical condition occurred. In the same line of the message, one of the following causes is indicated:

ACB ALREADY OPENED
UNSUPPORTED OPEN OPTIONS
INVALID OPEN MODE IN ACB
OPEN/RST – FILE NOT EMPTY

System Action: An ABEND and a full partition dump are forced.

AVBO100 ADABAS BRIDGE FOR VSAM OPENING FILE xxxxxxx

Explanation: AVB is active; the Adabas file corresponding to "xxxxxx" is processed.

System Action: The message is written to SYSLOG (for VSE/ESA only).

AVBO101 CRITICAL ERROR ON OPEN

Explanation: Internal logic error.

User Action: Contact your Software AG technical support representative.

AVBO102 INVALID OPEN/RST REQUEST

Explanation: An open for output was attempted for a file that is not empty, and the transparency table definition does not specify `RESET=DELETE` or `RESET=APPEND`.

AVBO103 OPEN/RST DELETE ERROR

Explanation: During delete processing for an OPEN/RST request, a nonzero Adabas response code was received.

AVBO104 ACB ADDRESS FOR FILE filename IS address

Explanation: The ACB for the specified file is allocated at the specified address.

AVBO105 ADDRESS FOR AREA avbbuffername IS address

Explanation: The specified AVB buffer is GETMAINed at the specified address.

AVBO106 ADDRESS FOR TAB tablename IS address

Explanation: The specified transparency table is loaded at the specified address.

AVBRnnn – Messages at Batch Request Time

These messages are written to the operator console.

AVBR098 REQUEST ERROR ADABAS RSP XXX STATUS YY

Explanation: It was not possible to set an equivalent VSAM status for the Adabas response code. The VSAM status is set to 92 and the Adabas RSP indicates the actual Adabas error.

AVBR099 INTERNAL ERROR

Explanation: An illogical condition occurred. In the same line of the message, one of the following causes is indicated:

ACB WAS NOT OPENED PREVIOUSLY
ACB DDNAME DOES NOT MATCH
FILE IS NOT OPEN
INVALID VSAM REQUEST CODE

System Action: An ABEND and a full partition dump are forced.

AVBR101 CRITICAL ERROR ON REQUEST

Explanation: Internal logic error.

User Action: Contact your Software AG technical support representative.

AVBnnnxx – Other Batch Messages

These messages are written to SYSOUT (OS/390, z/OS) or SYSLST (VSE/ESA).

AVB010*xx* ABEND Messages

Explanation: These messages accompany the debugging information produced by AVB when a TRACE 2, 3, or 5 is requested.

System Action: A dump of selected control blocks is produced.

User Action: Use the information provided along with any other information available (for example, program listing, assembly listing of the transparency table, log messages, Adabas command log, input to ADACMP) to identify and correct the problem.

AVB020*xx* Messages containing additional control card information

AVB02010 INVALID CONTROL CARD (MVS only)

Explanation: A syntax error occurred.

System Action: The program is terminated.

User Action: Correct the AVBUSER control card.

AVB030*xx* CRITICAL CLOSE ERROR (ABEND U0100 UNDER MVS)

Explanation: These messages are set in the ABEND/MSGS block before AVB forces an ABEND and a full dump because a critical error occurred during close.

System Action: An ABEND is forced and a full dump is produced.

User Action: Collect all information available (for example, printout of dump, program listing, assembly listing of transparency table, log messages, Adabas command log, input to ADACMP) and contact your Software AG technical support representative.

AVB040*xx* CRITICAL OPEN ERROR (ABEND U0100 UNDER MVS)

Explanation: These messages are set in the ABEND/MSGS block before AVB forces an ABEND and a full dump because a critical error occurred during open.

System Action: An ABEND is forced and a full dump is produced.

User Action: Collect all information available (for example, printout of dump, program listing, assembly listing of transparency table, log messages, Adabas command log, input to ADACMP) and contact your Software AG technical support representative.

AVB050*xx* CRITICAL REQUEST ERROR (ABEND U0100 UNDER MVS)

Explanation: These messages are set in the ABEND/MSGS block before AVB forces an ABEND and a full dump because a critical error occurred during processing.

System Action: An ABEND is forced and a full dump is produced.

User Action: Collect all information available (for example, printout of dump, program listing, assembly listing of transparency table, log messages, Adabas command log, input to ADACMP) and contact your Software AG technical support representative.

AVB070*xx* Messages from the common I/O module for internal AVB files

AVB110*xx* Messages issued with statistics printed at termination (OS/390 only)

AVBnnaa – CICS Messages

These messages occur in CICS environments during startup, shutdown, the AVB1 or AVB0 transactions, or the programs AVBCICSa.

The following messages appear at the terminal, and in some cases, on the console:

AVB0001 INVALID FUNCTION

Explanation: The function requested was not `ON`, `OFF`, `FILES`, or `STATS`.

System Action: None.

User Action: Correct the function code and press ENTER.

AVB0002 INVALID/MISSING PASSWORD

Explanation: The password for use of transaction AVB1 was incorrect.

System Action: None.

User Action: Correct the password and press ENTER.

AVB0003 UNABLE TO LOCATE AVBCICS2

Explanation: The program cataloged as AVBCICS2 is not the one provided on the AVB tape.

System Action: AVB cannot be activated.

User Action: Get a CSERV or DUMPT of the program and contact your Software AG technical support representative.

AVB0004 UNABLE TO LOCATE AVBCICS3

Explanation: The program cataloged as AVBCICS3 is not the one provided on the AVB tape.

System Action: AVB cannot be activated.

User Action: Get a CSERV or DUMPT of the program and call your Software AG technical support representative.

AVB0005 UNABLE TO LOCATE AVBCICS4

Explanation: The program cataloged as AVBCICS4 is not the one provided on the AVB tape.

System Action: AVB cannot be activated.

User Action: Get a CSERV or DUMPT of the program and call your Software AG technical support representative.

AVB0008 AVBRIDGE ALREADY ACTIVATED

Explanation: The `ON` function was requested but AVBRIDGE is already active.

System Action: AVB remains active.

User Action: None.

AVB0009 AVBRIDGE ALREADY DEACTIVATED

Explanation: The `OFF` function was requested but AVBRIDGE is already deactivated.

System Action: AVB remains deactivated.

User Action: None.

AVB000V AVBCICSV INSTALL VERIFICATION BEGINS...

Explanation: The program AVBCICSV invoked from AVB1, AVB0 (AVBCICS0) or at CICS initialization has begun processing.

System Action: None.

User Action: None.

AVB000V ADABAS VERSION IN AVBOPT IS n

Explanation: The value for the ADAVER (VERADA) parameter in the AVBOPT table is displayed.

System Action: None.

User Action: None, unless the Adabas version shown is not what was intended. Currently, "6" or "7" are the only valid ADAVER (VERADA) values. To change the ADAVER (VERADA) value, deactivate AVB, correct the MCOPT macro or the source AVBOPT module, and reassemble and link the AVBOPT module. Then reload AVBOPT into the CICS* environment and restart AVB. Do **not** use CEMT NEW to reload AVBOPT.

***** Because the module indicated is resident, loading it into CICS with CEMT NEW produces unpredictable results.

AVB000V ADABAS LINK EP–NAME IS: xxxxxxxx

Explanation: The value for parameter ADALNAM (ADALNKNM) in the MCOPT macro is displayed.

System Action: None.

User Action: None, unless the value shown is not what was intended. To change the ADALNAM (ADALNKNM) value, deactivate AVB, correct the value in the MCOPT macro or the source AVBOPT module, and reassemble and link the AVBOPT module. Then reload AVBOPT into the CICS* environment and restart AVB. Do **not** use CEMT NEW to reload AVBOPT.

***** Because the module indicated is resident, loading it into CICS with CEMT NEW produces unpredictable results.

AVB0011 FILE NOT FOUND

Explanation: The file specified in the "OPTION" field was not found in AVBTAB.

System Action: The update function cannot be performed.

User Action: Verify the spelling of the file name and its presence in AVBTAB.

AVB0012 ERRORS DETECTED DURING UPDATE

Explanation: An error was detected during UPDATE function. One or more file names were not found in the FCT or the DBID was invalid.

System Action: Only valid updates are performed.

User Action: Verify the validity of the fields to update and retry.

AVB0013 INVALID SOFTWARE ENVIRONMENT

Explanation: The release or version of one or more of the software components is not supported by AVB.

System Action: AVB cannot be installed.

User Action: Verify the system requirements.

AVB0014 ADABAS LINK EP–NAME NOT FOUND IN PPT

Explanation: The value for the ADALNKNM parameter of the MCOPT macro either does not correspond with an Adabas link routine name in the PPT or the link routine was not found in the load list.

System Action: AVB cannot be activated. If the AVB1 transaction is being executed, abend AVBV occurs.

User Action: Check the ADALNKNM in the MCOPT macro, check the PPT or the libraries concatenated in the CICS job control. If necessary, contact your Software AG technical support representative.

AVB0014 PROG xxxxxxx NOT FOUND IN PPT

Explanation: Any of the AVB programs were not included in the CICS PPT.

System Action: AVB cannot be installed.

User Action: Verify the entries required in the PPT.

AVB0015 UNABLE TO LOCATE AVBOPT

Explanation: The program cataloged as AVBOPT is not a valid options table.

System Action: AVB cannot be activated.

User Action: Verify the assembly listing of AVBOPT and make sure it is free of errors. If necessary, contact your Software AG technical support representative.

AVB0016 UNABLE TO LOCATE AVBPROG

Explanation: The program cataloged as AVBPROG is not the one provided in the AVB tape.

System Action: AVB cannot be activated.

User Action: Get a CSERV or DUMPT of the program and contact your Software AG technical support representative.

AVB0017 UNABLE TO LOCATE AVBTAB

Explanation: The program cataloged as AVBTAB is not a valid transparency table.

System Action: AVB cannot be activated.

User Action: Verify the assembly listing of the transparency table and make sure that it is free of errors. If the problem persists, contact your Software AG technical support representative.

AVB0018 VSAM FILE xxxxxxxx NOT FOUND IN FCT

Explanation: Every entry in the transparency table must have a corresponding entry in the CICS FCT, even when the VSAM file no longer exists.

System Action: No request against that file is processed.

User Action: Provide the adequate entry in the CICS FCT.

AVB0020 FILE xxxxxxxx OPENED BY CICS

Explanation: This message is issued to the operator console during CICS initialization when executing program AVBCICS0 from PLT. A file whose entry in AVBTAB has the parameter OCIND set to "O" (open) was found and opened by CICS. (This message is issued only by program AVBCICS0.)

System Action: The file is set closed to AVB.

User Action: Close the file using the CEMT transaction and open it using AVB1 transaction.

AVB0024 SETXIT CODE HAS BEEN ENTERED– PROGRAM CHECK DURING INSTALLATION

Explanation: AVB SETXIT took control because of a program check during installation.

System Action: AVB cannot be installed.

User Action: Print the CICS dump file. Locate the dump caused by ABEND AVHO and contact your Software AG technical support representative.

AVB0025 ONE OR MORE FILES NOT IN FCT

Explanation: A request for the function FILES with the option OPEN found one or more files without a valid entry in the FCT.

System Action: Files without an entry in the FCT are set closed.

User Action: Include the necessary entries in the CICS FCT.

AVB1000 ADABAS BRIDGE FOR VSAM – V5.1.s– date {OS/390 MVS | VSE/ESA} v.r *—*CICS/ESA v.r – ***** ADABAS BRIDGE FOR VSAM ACTIVATED *****

Explanation: AVB was successfully installed.

System Action: VSAM requests for files under AVB are converted to Adabas calls.

User Action: None.

AVB100V INCORRECT VERSION FOR MODULE: xxxxxxxx

Explanation: The indicated module is not valid for the current version of AVB (V5.1.s).

System Action: AVB cannot be activated. If transaction AVB1 is running, ABEND AVBV occurs.

User Action: Obtain a CSERV or DUMPT of the indicated module, check the PPT, the AVBLINKC job, or the CICS library concatenation. Contact your Software AG technical support representative for assistance.

AVB102V ADABAS VERSION AVBOPT VERSION MISMATCH

Explanation: The `VERADA` parameter in AVBOPT indicates Adabas version 6, but AVB has determined either that the link routine is not valid for version 6 or that the Adabas Migration Aids are not installed.

System Action: AVB cannot be activated. If transaction AVB1 is running, abend AVBV occurs.

User Action: Check the version in the MCOPT macro. If correct, check the Adabas link routine version (ADALNC) or the Migration Aids. Contact your Software AG technical support representative for assistance.

AVB104V MODULE: xxxxxxxx NOT FOUND IN MODULE TABLE

Explanation: An internal error has occurred.

System Action: AVB cannot be activated. If transaction AVB1 is running, ABEND AVBV occurs.

User Action: Obtain a dump and contact your Software AG technical support representative for assistance. If the error occurred during CICS initialization, use the AVB1 transaction to obtain the error and the dump.

AVB105V VERSION FOR MODULE: xxxxxxxx NOT VALID

Explanation: An internal error has occurred.

System Action: AVB cannot be activated. If transaction AVB1 is running, ABEND AVBV occurs.

User Action: Obtain a dump and contact your Software AG technical support representative for assistance. If the error occurred during CICS initialization, use the AVB1 transaction to obtain the error and the dump.

AVB9000 ADABAS BRIDGE FOR VSAM TERMINATED

Explanation: This is the normal termination message issued by the AVB shutdown program.

AVB9999 AVBRIDGE NOT INSTALLED

Explanation: AVB was not installed because of errors detected and reported previously (AVBCICS0).

System Action: AVB was not installed.

User Action: A previous message explains the reason. Correct the error and try again.

AVB999V AVBRIDGE CICS INSTALLATION FAILS

Explanation: Program AVBCICSV has detected one or more errors.

System Action: AVB cannot be activated. If the AVB1 transaction was running, ABEND AVBV occurs.

User Action: Check the operator console for previous message(s) explaining the cause of the error. Contact your Software AG technical support representative if necessary.

AVB99AV AVBCICS WORKAREA ADDRESS INVALID

Explanation: An internal error has occurred.

System Action: AVB cannot be activated. If the error occurred during the AVB1 transaction, an AVBV ABEND occurs.

User Action: If the error occurred during CICS initialization or from the AVB0 transaction, submit the AVB1 transaction to activate AVB, obtain the CICS dump, and contact your Software AG technical support representative.

AVB CICS Transaction Dump IDs

The codes in this section identify the snapshot dumps that AVB writes to the CICS transaction dump dataset when an AVB-related error occurs under CICS. Some conditions may cause CICS to terminate abnormally.

If you need to call Software AG technical support for assistance, it is important to have the following information available: the printouts of the CICS dump, the CICS JCL execution, and the Adabas command log; the assembly listing of the transparency table; the input to ADACMP; the VSAM record layout; and, if possible, the assemble/compile listing of the application program.

AVA4

Explanation: The Adabas command A4 failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVBA

Explanation: An internal error in FCT processing occurred.

User Action: Contact your Software AG technical support representative.

AVBB

Explanation: An internal error occurred. Open any close indicators that are not properly set.

User Action: Contact your Software AG technical support representative.

AVBF

Explanation: A transaction is trying to use more than 100 files.

User Action: Limit the transaction to use a maximum of 100 files.

AVBG

Explanation: An invalid request code was specified.

User Action: Verify that the request passed to AVB is a supported request.

AVBH

Explanation: An internal error occurred. The activity table is full.

User Action: Contact your Software AG technical support representative.

AVBI

Explanation: A transaction is trying to use more than one DBID.

User Action: Verify that all files used by the transaction are in the same database.

AVBJ

Explanation: A GETMAIN for an I/O area failed.

User Action: Verify the availability of storage. Contact your Software AG technical support representative if necessary.

AVBK

Explanation: An internal error occurred. VSAM feedback is not set.

User Action: Contact your Software AG technical support representative.

AVBL

Explanation: An internal error occurred; an invalid function was specified.

User Action: Contact your Software AG technical support representative.

AVBP

Explanation: The PCT entry has a TWASIZE of less than 24 bytes. If that is not the case, it is probably a new CICS version.

User Action: Verify that all transactions using AVB have at least 24 bytes. The contents of these 24 bytes are saved before being used by AVB and restored after calling Adabas.

AVBS

Explanation: A GETMAIN for shared storage failed.

User Action: Verify the availability of storage. Contact your Software AG technical support representative if necessary.

AVBT

Explanation: An internal error occurred during a backout command.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVIT). Contact your Software AG technical support representative if necessary.

AVBV

Explanation: The program AVBCICSV invoked by the AVB1 transaction has detected one or more errors.

User Action: Check the operator console for previously issued messages indicating the cause of the error. Contact your Software AG technical support representative if necessary.

AVB1

Explanation: A GETMAIN for terminal storage failed.

User Action: Verify the availability of storage. Contact your Software AG technical support representative if necessary.

AVCL

Explanation: An internal error occurred while processing a close.

User Action: Contact your Software AG technical support representative.

AVDA

Explanation: An internal error occurred during a delete. An attempt to delete was made specifying a key that is not unique.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVDB

Explanation: An internal error occurred during an update.

User Action: Contact your Software AG technical support representative.

AVDU

Explanation: An internal error occurred while reading an alternate index file.

User Action: Contact your Software AG technical support representative.

AVDV

Explanation: An internal error occurred during the processing of duplicate keys.

User Action: Contact your Software AG technical support representative.

AVD2

Explanation: An internal error occurred during a delete. Unexpected error after L9 Adabas command.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVD8

Explanation: An internal error occurred during an update. An attempt was made to update keys that are not unique.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVET

Explanation: An Adabas command ET failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVEU

Explanation: An internal error occurred while processing an open command. There is an invalid file list in the format buffer.

User Action: Contact your Software AG technical support representative.

AVE4

Explanation: An Adabas command E4 failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVGT

Explanation: A SETL request with option GT failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVHI

Explanation: An Adabas command HI failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVHO

Explanation: A program check was detected when executing AVBCICS0 or AVBCICS1.

User Action: See the message before the ABEND.

AVHR

Explanation: A nonzero Adabas response occurred while processing the L6 command on a multiple-record-type file.

User Action: Contact your Software AG technical support representative.

AVID

Explanation: A generic delete (not supported) was requested, or a delete/update was requested but not allowed by the CICS file control table.

User Action: Correct the CICS FCT and/or transaction.

AVIE

Explanation: An ESETL was requested but was not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVIF

Explanation: A SETL was requested but was not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVIG

Explanation: A GET was requested but either the request or the options were not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVII

Explanation: An INSERT was requested but was not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVIL

Explanation: An error occurred during sequential processing. The record length is greater than specified in the transparency table.

User Action: Correct the transparency table to specify the maximum record length.

AVIM

Explanation: A GETMAIN for a FWA was rejected due to incompatible options.

User Action: Correct the transaction.

AVIN

Explanation: An internal error occurred during an insert. Unexpected error after Adabas L9 command.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVIP

Explanation: The PUT/DELETE requested was not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVIR

Explanation: An internal error occurred during sequential processing.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVIS

Explanation: A request was made involving use of segmented records. (Segmentation is not supported.)

User Action: Correct the application.

AVIU

Explanation: The UPDATE requested was not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVL1

Explanation: An Adabas command L1 failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVMR

Explanation: Nonzero Adabas response processing L3 command for record ID outside of key range or generic key.

User Action: Contact your Software AG technical support representative.

AVNX

Explanation: An error occurred during sequential processing.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVN1

Explanation: Adabas command N1 failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVOP

Explanation: An internal error occurred while processing an open command.

User Action: Contact your Software AG technical support representative.

AVP1

Explanation: An internal error occurred.

User Action: Contact your Software AG technical support representative.

AVP2

Explanation: An error occurred during sequential processing. {GET | READ} NEXT request without previous explicit or implicit start.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVP3

Explanation: A request for an insert, update, or delete was made, but no modification is allowed for the file.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVRB

Explanation: An internal error occurred while processing an L9 command.

User Action: Contact your Software AG technical support representative.

AVRC

Explanation: The Adabas command RC failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVRD

Explanation: An internal error occurred during a direct read. An unexpected error occurred after an Adabas L3 command.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVRH

Explanation: An internal error occurred during a direct read for an update. An unexpected error occurred after an Adabas L6 command.

User Action: Get Adabas response code and debugging information from the AVBCICS5 trace table. Contact your Software AG technical support representative if necessary.

AVRI

Explanation: An Adabas command RI failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVR2

Explanation: An update was requested, but there is no record with the specified key.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVSA

Explanation: An Adabas command S1/S4 failed.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information on the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVST

Explanation: An error occurred after executing an Adabas command L9.

User Action: Use the AVB trace table (transaction AVTR) and get the Adabas response code and other information required to identify the problem. Use the information in the CICS dumps produced by the ABEND; there are dumps of the dynamic areas, the AVB trace table (code AVTR), the activity table (code AVAT), and the transparency table (code AVTT). Contact your Software AG technical support representative if necessary.

AVS2

Explanation: An internal error occurred.

User Action: Contact your Software AG technical support representative.

AVT1

Explanation: An internal logic error occurred.

User Action: Contact your Software AG technical support representative.

AVTC

Explanation: An undefined record type field was specified. The record type field contains values not specified either in the transparency table or in the default record IDG.

User Action: Contact your Software AG technical support representative if necessary.

AVUU

Explanation: The key length specified is larger than 125 bytes.

User Action: Limit the key length to a maximum of 125 bytes.

AVWL

Explanation: A GETMAIN for VSWA was rejected because of incompatible options.

User Action: Correct the transaction.

AVWN

Explanation: The {GET | READ} NEXT requested was not allowed by the CICS FCT.

User Action: Correct the CICS FCT and/or transaction.

AVWR

Explanation: An internal error occurred. An invalid request was made after calling the AVB I/O analyzer.

User Action: Contact your Software AG technical support representative.

AVX3

Explanation: An internal error occurred while processing “record type” files.

User Action: Contact your Software AG technical support representative.

AVX5

Explanation: The key length specified is larger than 125 bytes.

User Action: Limit the key length to a maximum of 125 bytes.

AVX6

Explanation: The key length specified is larger than 125 bytes.

User Action: Limit the key length to a maximum of 125 bytes.

AV61

Explanation: A GETMAIN for terminal storage failed.

User Action: Verify availability of storage.

AV62

Explanation: An error occurred locating the trace table.

User Action: Verify that AVB was initialized and is active. Contact your Software AG technical support representative if necessary.

AV63

Explanation: An error occurred while locating AVBFIXTB.

User Action: Verify the PPT entry for program AVBFIXTB. Contact your Software AG technical support representative.

AV71

Explanation: A GETMAIN for terminal storage failed.

User Action: Verify the availability of storage.

AV81

Explanation: A GETMAIN for terminal storage failed.

User Action: Verify the availability of storage.

AV82

Explanation: An internal error occurred.

User Action: Contact your Software AG technical support representative.

AV83

Explanation: An error occurred while locating the options table (AVBOPT).

User Action: Verify that AVB is installed and active for CICS. Contact your Software AG technical support representative if necessary.

AV90

Explanation: An error occurred while locating AVBCICS2 during termination processing.

User Action: Verify that AVB was installed properly for CICS. Contact your Software AG technical support representative if necessary.

ET User Response Codes

The following response codes refer to the ET, BT, RE, and C1 functions associated with the call to the AVBETBT entry point when running batch programs as an ET user:

0xxx ADABAS RESPONSE CODE AS SPECIFIED BY THE VALUE

Explanation: This is one of approximately 200 Adabas response codes.

User Action: Consult the *Adabas Messages and Codes Manual* for an explanation of the response code.

1001 AVBRIDGE MODULE NOT PROPERLY LOADED

Explanation: An installation or JCL/JCS error has probably occurred.

User Action: Check that the proper load or core image libraries have been specified in the JCL/JCS. Ensure that version 5.1.s of AVB is installed and active.

1002 AVBUSER MODE=ET CARD NOT GIVEN

Explanation: The user did not specify `AVBUSER MODE=ET` in the AVB control cards for that job execution.

User Action: Specify `AVBUSER MODE=ET` and reexecute.

1003 INVALID AVBETBT FUNCTION SPECIFIED

Explanation: The application program is passing an incorrect function (that is, a function other than ET, SD, BT, RE, or C1).

User Action: Correct the application program to pass the proper function and reexecute.

1004 CDLOAD FOR AVBPVT MODULE FAILED

Explanation: An installation error has occurred or AVB is not active.

User Action: Check the AVB installation and activation procedures for VSE. Check the JCS used.

1005 OPTIONS TABLE AVBOPT NOT LOADED

Explanation: The AVB options table (AVBOPT) was not loaded.

User Action: Ensure that version 5.1.s of AVB is executing and that it has been initialized properly.

1006 INVALID ETBT–INFO PARM GIVEN

Explanation: The pointer to the ET user ID / ET data length field was not passed.

User Action: Modify the application program to pass four parameters, ensuring they are positionally correct.

1007 ET DATA LENGTH INVALID

Explanation: The ET data length field contains an invalid value: either a value of zero for the RE function or a value greater than 2000 for any function that uses the field.

User Action: Check the application program to ensure that the proper ET data length is supplied.

1008 INVALID ETBT–WORKAREA PARM GIVEN

Explanation: The pointer to the ET data work area was not passed.

User Action: Check the application program to ensure that {four | five} parameters are passed and that they are positionally correct.

1009 INCORRECT AVB ENVIRONMENT

Explanation: The pointer to AVBVCONS was not initialized.

User Action: Ensure AVBINIT worked successfully. Look for messages on SYSLOG. Check that AVBTAB was loaded.

AVBnnn – Messages from AVB Online Services

When you receive a message from AVB online services, enter the direct command

INFO

—to display an explanation of the message together with a recommended action.

Entering the direct command

INFO nnn

—displays the long description of the AVB online services message “nnn”.

A

APPENDIX A—DEMONSTRATION FILES

The chapters **Defining the Adabas File Structure** and **Creating the Transparency Table** use the following COBOL file definitions to illustrate Adabas field definitions and transparency table parameters, respectively.

EMPLOYEES File

```
FD EMPLOYEES
   LABEL RECORDS ARE STANDARD
   DATA RECORD IS EMPL-REC.

01 EMPL-REC.
   05 EMPL-NAME          PIC X(32).
   05 EMPL-SSNO          PIC 9(09).
   05 EMPL-DEPT          PIC X(05).
   05 EMPL-TITLE         PIC X(15).
   05 EMPL-PHONE         PIC X(04).
```

INVOICES File

```
FD INVOICES
   LABEL RECORDS ARE STANDARD
   DATA RECORD IS INVOICE.

01 INVOICE.
   05 INV-NO             PIC X(8).
   05 CUST-NO            PIC S9(7) COMP-3.
   05 FNAME              PIC X(20).
   05 LNAME              PIC X(25).
   05 ITEM-CNT           PIC 9(3).
   05 ITEMS OCCURS 1 TO 150 TIMES DEPENDING ON ITEM-CNT.
      10 ITEM-NO         PIC 9(5)
      10 ITEM-QTY        PIC 9(3).
      10 ITEM-PRC        PIC 9(6).
```

STUDENTS File

```
FD STUDENTS
   LABEL RECORDS ARE STANDARD
   DATA RECORDS ARE STUDENT-INFO STUDENT-ACCT.

01 STUDENT-INFO.
   05 KEY.
      10 ID-NO             PIC X(6).
      10 RECTYPE           PIC X(1).
   05 FULL-NAME.
      10 FNAME             PIC X(15).
      10 LNAME             PIC X(20).
   05 ADDRESS.
      10 STREET            PIC X(25).
      10 CITY              PIC X(20).
      10 STATE             PIC X(2).
      10 ZIP               PIC 9(9).
   05 DEPT                 PIC X(4).
   05 ADVISORS             PIC X(20) OCCURS 2 TIMES.

01 STUDENT-ACCT.
   05 KEY.
      10 ID-NO             PIC X(6).
      10 RECTYPE           PIC X(1).
   05 TERM                 PIC X(6).
   05 CHARGES.
      10 TUITION           PIC 9(6).
      10 ROOM              PIC 9(5).
      10 MEALS             PIC 9(5).
   05 PMTS                 PIC 9(1).
   05 PAYMENTS OCCURS 1 TO 3 TIMES DEPENDING ON PMTS.
      10 DATE              PIC 9(6).
      10 AMOUNT            PIC 9(6).
      10 MEMO              PIC X(40).
```

B

APPENDIX B— SAMPLE ENTRIES FOR ESDS AND RRDS FILES

When using ESDS or RRDS files, follow the same installation and verification procedures as for other VSAM files, but substitute the modules listed on page 20 during the steps for loading test files and verifying the installation.

AVB uses the VSAM relative record number (RRN) as the Adabas ISN. For RRDS files, you must build your own VSAM RRN and will need to use an Adabas user exit 6 in the ADACMP compression step when loading the file. See page 18.

Add FCT entries for the test files you load. Supported file entries are listed on page 35.

The parameter VSAMTYP must be coded in the transparency table (MCTAB) if you are using ESDS or RRDS files and the PREFSZE parameter must be set to zero (0). See page 104.

Sample entries for ESDS and RRDS files are provided in this appendix.

Example Entries for RRDS Files (COBOL)

Sample FCT entries for the file SYS021R (COBOL):

```
SYS021R DFHFCT TYPE=DATASET,DATASET=SYS021R,                    X
        ACCMETH=(VSAM,RRDS),                                    X
        SERVREQ=(GET,UPDATE,NEWREC,BROWSE,DELETE,PUT),          X
        RECFORM=(VARIABLE,BLOCKED),                             X
        FILSTAT=(ENABLED,CLOSED),BUFNI=4,BUFND=4,               X
        LSRPOOL=NONE,RSL=PUBLIC,STRNO=3,LOG=NO
```

The COBOL file definition below defines the record as it appears to the COBOL program:

```
FD DEMOFILE
   LABEL RECORDS ARE STANDARD.
   01 DEMO-REC.
   05 DEMO-SSNO                 PIC 9(04).
   05 DEMO-NAME                 PIC X(25).
   05 DEMO-DEPARTMENT           PIC X(05).
```

The following transparency table entry defines the file to AVB:

```
MCTAB    TYPE=GEN,FN=DEMO,FNR=21,RECSIZ=34,VSAMTYP=RRDS,
         FORMAT='GA'
```

—where

FN	is the VSAM DDNAME or DTFNAME (DEMO) referenced by the program
FNR	is the Adabas file number (21)
RECSIZ	is the physical record size (34)
VSAMTYP	is the VSAM dataset type (RRDS)
FORMAT	is the Adabas format buffer description that will be used to retrieve the record (GA).

The following ADACMP definitions describe the file to Adabas:

```
ADACMP COMPRESS
ADACMP FNDEF='01,GA'           DEMO-REC
ADACMP FNDEF='02,AA,04,U'      DEMO-SSNO
ADACMP FNDEF='02,AB,25,A'      DEMO-NAME
ADACMP FNDEF='02,AC,05,A'      DEMO-DEPARTMENT
```

Example Entries for ESDS Files (PL/I)

Sample FCT entries for the file PLIESDS (PL/I):

```
PLIESDS DFHFCT TYPE=DATASET,DATASET=PLIESDS,                    X
        ACCMETH=(VSAM,ESDS),                                    X
        SERVREQ=(GET,UPDATE,NEWREC,BROWSE,DELETE,PUT),          X
        RECFORM=(FIXED,BLOCKED),                                X
        FILSTAT=(ENABLED,CLOSED),BUFNI=4,BUFND=4,               X
        LSRPOOL=NONE,RSL=PUBLIC,STRNO=3,LOG=NO
```

The PL/I file definition below describes the record as it appears to the PL/I program:

```
FD DEMOFILE
   LABEL RECORDS ARE STANDARD.
   01 DEMO-REC.
   05 DEMO-NAME               PIC X(25).
   05 DEMO-DEPARTMENT         PIC X(05).
   05 DEMO-TELEPHONE-NO       PIC X(15).
```

The following transparency table entry defines the file to AVB:

```
MCTAB    TYPE=GEN,FN=DEMO,FNR=22,RECSIZ=45,VSAMTYP=ESDS,
         FORMAT='GA'
```

—where

FN	is the VSAM DDNAME or DTFNAME (DEMO) referenced by the program
FNR	is the Adabas file number (22)
RECSIZ	is the physical record size (45)
VSAMTYP	is the VSAM dataset type (ESDS)
FORMAT	is the Adabas format buffer description that will be used to retrieve the record (GA).

The following ADACMP definitions describe the file to Adabas:

```
ADACMP COMPRESS
ADACMP FNDEF='01,GA'           DEMO-REC
ADACMP FNDEF='02,AA,25,A'      DEMO-NAME
ADACMP FNDEF='02,AB,05,A'      DEMO-DEPARTMENT
ADACMP FNDEF='02,AC,15,A'      DEMO-TELEPHONE-NO
```

INDEX

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

T

U

V

X