BrightStor™ CA-ASM2® Backup and Recovery

Planning Guide

4.2

Computer Associates™

Contents

Chapter 1. Introduction

This publication is designed for DASD Storage Administrators and other personnel responsible for the planning and implementation of CA-ASM2. The first chapter of this document outlines all of the general pre-installation considerations. Many decisions need to be made to customize CA-ASM2 to operate in your unique environment.

After reading Chapter 2, “General Planning,” you need to read Chapter 3, “SMP/E Installation Planning.” It is during this phase that the DASD Storage Administrator or other decision-making individual(s) makes decisions and enters the appropriate parameters on the worksheets provided in the Appendix. **The worksheets must be filled out before actually doing the installation.** The individual(s) making the decisions may then give the worksheets to the person doing the installation. This may be a system programmer or other person not involved in the decision-making process. With the worksheets filled out ahead of time, the installation proceeds in a timely fashion.

The installation phase is divided into three parts; the base product, IXR, and Allocation Manager installations. All may be installed at once or IXR and Allocation Manager can be installed at a later time.

Note: The *CA-ASM2 Getting Started* guide should be read to gain an overview of the entire product and to determine which of the optional components is desired.

1.1 Summary of New Features for Version 4.2

Each of the following Version 4.2 enhancements for CA-ASM2 are described in detail in the new set of documentation issued for this version.

1.1.1 9999 Files Tape Support

CA-ASM2 now supports up to 9,999 files on a single CA-ASM2 Archive or Backup tape.

- Allows unload to tape to hold 9999 files
- $FORMAT automatically updates LOxxx file(s)
- Converts old format LOxxx files to new format

1.1.2 SMS Support Enhancements

The Management Class fields controlling expiration may now optionally be used to control when CA-ASM2 will expire, archive and backup versions. In addition, CA-ASM2 invokes the ACS class selection routines prior to reload to handle situations in which the ACS class selection rules have changed and the data set is assigned to different classes.

- Reload pre-drives ACS routines to determine SMS classes during dynamic allocation
- Uses SMS MGTCLASS for retention values during archives and backups

1.1.3 ISPF Interface Enhancements

The CA-ASM2 ISPF application has been enhanced to provide additional functionality to end users of CA-ASM2. Online panels are provided to perform common functions such as requesting the restore of all data sets for an application through a new Application construct and restoring a volume from a specific point in time. Customization capabilities are built in through the use of a profile, which removes options that are inappropriate for general users.

- Group data sets at the application level
- High-level qualifier masking and wild cards

1.1.4 IXR Enhancements

The temporary name created by IXR to restore a data set now supports the specification of up to three alias levels for non-VSAM data sets and two levels for VSAM data sets.

Also a New RLDTMPNM PARMLIB keyword for data set high-level qualifier usage during reload UCAT processing has been added.

1.1.5 CA-ASM2 Workstation

CA-ASM2 integration with the CA-ASM2 Workstation has been enhanced to provide extensive administration, reporting, and storage management capabilities from a Windows-based graphical user interface. CA-ASM2 Workstation is a separately licensed product.

1.1.6 $DASDMNT

Support for SMS management class retention of archived data sets has been implemented via the use of a new keyword $SMSRTPD which can be coded in the SYSIN stream.

1.2 Documentation Changes

- A new document, the *ISPF User Interface Guide* has been added to the documentation set. This document describes all the new extended features developed for Version 4.2. In addition, all ISPF panels that were previously described in Appendix A of the *CA-ASM2 System Reference Guide* and Appendix A of the *CA-ASM2 RSVP User Guide* have been integrated into this document so that a single source can be referenced for all ISPF applications.
- All guides have been updated with relevant Version 4.2 information.
- The syntax throughout the documentation has been given a new look. Information has been provided to assist you in reading the syntax diagrams.
- The Troubleshooting section, found in the *CA-ASM2 System Reference Guide*, now provides information for accessing the Computer Associates home page on the Internet for additional Computer Associates products and services.

1.2.1 Removed

- *CA-ASM2 General Information Guide*. Information located in this guide has been disseminated into the *CA-ASM2 System Reference Guide* and *CA-ASM2 Getting Started*.
- Conversion guides for CA-3 and CA-Dynam/DASD.
- *CA-ASM2 Master Index*.
- Demand Analysis Request (DAR) form. You can now enter your request through StarTCC Extended Support (click on Support at www.ca.com on the Web).

Chapter 2. General Planning

This chapter documents the various planning considerations that should be made by the Storage Administrator or other decision making individual(s) before installation and customization of CA-ASM2. By completing the worksheets described in this chapter, the installation of the base product is simplified and relatively straightforward.

Planning considerations that are applicable to an installation are described in this chapter. Specific installation variables and parameters are described in the following chapter. You need to read Chapter 3, “SMP/E Installation Planning” and complete the associated worksheets.

This guide contains everything you need to know in planning the installation of CA-ASM2 with SMP/E. While the product is easy to install, it is also a powerful and versatile system with options and facilities you can default or change to meet the requirements of your installation. This chapter will help you make these decisions.

Note: All references to MVS are applicable to MVS/XA and MVS/ESA unless otherwise noted.

2.1 Installing CA-ASM2

CA-ASM2 Version 4.2 is installed using SMP/E. The machine-readable program materials required for installation are distributed as a multifile installation tape in SMP/E format. For a general description of the tape format, see the heading titled "Installation Materials."

SMP/E may be used to perform product installation, tailoring, and maintenance. It provides the ability to control these activities in a consistent manner.

There are three major operations performed by SMP/E when installing a product or performing maintenance. These operations manage a structure wherein a given product is present in two places: distribution libraries and target (operating) libraries. The distribution libraries are used for maintenance operations only. The product executes from the target libraries.

The operations performed are as follows:

- RECEIVE Processing

  The installation tape is loaded by SMP/E into temporary data sets. If any error is detected, or the user wishes to stop the process at this point, a REJECT operation may be run, which undoes anything done during RECEIVE processing.

- APPLY Processing

  During APPLY processing, SMP/E performs the operations dictated by the modification control statements (MCS), and updates the target libraries. You may then test the modification. You should complete the installation prior to testing any components. If the installation is to be aborted at this point, a RESTORE operation may be run, which restores the system libraries from the distribution libraries.

- ACCEPT Processing

  When the ACCEPT operation is run, the modification is permanently placed in the distribution libraries. There is no direct way to undo the modification once an ACCEPT has been run.

2.2 CA-ASM2 and CAIRIM

For Version 4.2 of CA-ASM2, both SMP/E and CAIRIM are required to install the product. The Computer Associates Resource Initialization Manager (CAIRIM) utility is executed to dynamically install modules and SVCs into the system's Link Pack Area (LPA) and is also used to install MVS subsystems. CAIRIM is available only on the CA Common Services (formerly Unicenter TNG Framework for OS/390) tape shipped with the current versions of most Computer Associates products.

CAIRIM is additionally used to install system modifications. Three of the five possible IXR system modifications are installable with CAIRIM, while all of the three possible Allocation Manager modifications are installable with CAIRIM. The four system modifications for the base CA-ASM2 component are packaged as SMP/E USERMODs.

CAIRIM must be installed from the CA Common Services tape prior to installing IXR or Allocation Manager system modifications, however it is best to install CAIRIM prior to installing the CA-ASM2 base component. The planning in this document provides information on sharing target libraries with CA Common Services components and should be read prior to beginning the CA Common Services installation. The *CA Common Services Getting Started* guide provides the instructions on how to install CAIRIM; these guides contain materials specific to installing CA-ASM2 system modifications with CAIRIM.

2.3 CA Common Services

The CA Common Services are a group of system services that protect your investment in software solutions by helping you manage your data center more efficiently. Each of the CA Common Services offers individual benefits to the user. The CA Common Services which are used with, and benefit CA-ASM2 are described next.

2.3.1 CA LMP

The Computer Associates License Management Program (CA LMP) provides a standardized and automated approach to the tracking of licensed software. It uses common realtime enforcement software to validate the user's configuration. CA LMP reports on activities related to the license, usage and financials of Computer Associates solutions. The routines which accomplish this are integrated into the Computer Associates MVS dynamic service code, S910 (the CAIRIM service). CA LMP features include:

- Common Key Data Set can be shared among many CPUs
- "Check digits" are used to detect errors in transcribing key information
- Execution Keys can be entered without affecting any Computer Associates software solution already running
- No special maintenance requirements

2.3.2 CAIRIM

CAI Resource Initialization Manager (CAIRIM) is the common driver for a collection of dynamic initialization routines that eliminate the need for user SVCs, SMF exits, subsystems, and other installation requirements commonly encountered when installing systems software. These routines are grouped under the Computer Associates MVS dynamic service code, S910. CAIRIM features include:

- Obtaining SMF data
- Verification of proper software installation
- Installation of MVS interfaces
- Automatic startup of Computer Associates and other vendor products
- Proper timing and order of initialization

2.4 System Requirements

This section provides an overview of the system requirements for the following:

- Operating System Interfaces
- Aging
- Operating System
- SMS

2.4.1 Operating System Interfaces

Your operating system environment may require some modifications before you can use all CA-ASM2 features. To help you determine what changes to make, read the summary that follows for each CA-ASM2 feature listed next. All of the listed features can be installed with SMP/E.

- Password Checking (optional)
- OPEN Modification (optional)
- CLOSE Modification (optional)
- TSO Command Authorization (required)
- MVS APF List (required)
- MVS Link List Option

2.4.1.1 Password Checking

This feature is optional, but highly recommended to enhance product usability. If archive, backup, and migrate facilities are to process password-protected data sets, the operator is prompted to supply the appropriate password. Add the names $DASDMNT, $MIGRATE, $MCLMVR, $GETAX, $COPYTP, and $VLIST (only for VSAM) to the MVS Program Properties table (often referred to as the PPT or IEFSDPPT), with attributes set to X'02' to bypass password checking. With MVS/XA 2.2 and later, the PPT entries are defined in a SCHEDxx member of SYS1.PARMLIB. Member PPTSAMP3 of SAMPJCL contains the SCHEDxx entries for MVS/XA 2.2 and later users. See the IBM MVS publications for the procedure to add programs to this table.

Member APPT$ of SAMPJCL contains further information on modifying the PPT. See the applicable IBM MVS publication for the procedure to add programs to the PPT.

In all cases, CA-ASM2 preserves the password protection bits in the DSCB to maintain data security.

2.4.1.2 OPEN Modification

CA-ASM2 provides an SMP/E-format modification to the IFG0196W CSECT of IBM module IGC0001I to maintain aging information in the Format-1 DSCB in the VTOC. The source of the modification is member L1OMVS6W of data set CAI.PPOPTION. This modification is compatible with SU60 which is provided with all current versions of MVS. If you are currently using SU60 to maintain aging information, you need not install the CA-ASM2 modification, however, the CA-ASM2 modification provides more information than SU60. You can install the CA-ASM2 OPEN modification regardless of whether you are currently using SU60. See 2-8 for more information on aging.

Note: The OPEN aging modification has been changed to provide support for SMS installations. This change was made available in a Version 3.1 APAR tape (volser A89299). If the OPEN modification has been installed from the APAR tape, you do not need to reinstall it when installing CA-ASM2, Version 4.2.

DFP 2.4 or later users must install the new OPEN aging modification if the version has not been installed from the APAR tape.

2.4.1.3 CLOSE Modification

CA-ASM2 provides an SMP/E format modification to the VSAM CLOSE CSECT IDA0192C of IBM module IDA0192A, so you can maintain and use aging information about VSAM clusters whether cataloged in a VSAM (non-ICF) catalog or under ICF (Integrated Catalog Facility). The source of the modification is member L1CMVS2C of data set CAI.PPOPTION. This modification maintains date-last-used, use count, and an SU60-like change flag. See 2-8 for more information on aging. This feature is optional.

Note: Users with DFP Version 3.1 and later need not install any VSAM aging modification because additional aging information (last usage date) is provided with DFP at these levels.

2.4.1.4 TSO Command Authorization

This feature is required if you intend to use the space management and unload/reload facilities of CA-ASM2. The following CA-ASM2 online commands require authorization:

$RA - Reload from Archive
$RB - Reload from Backup
$RC - Reload from Backup (alias for $RA)
$RL - Reload from Archive (alias for $RA)
$RS - Alias for $RSVP command
$RSVP - Main CA-RSVP command
$SM - Space Management: release unused space, compress, and so forth.

To authorize these commands, perform the following steps:

1. If TSO/E is installed, authorize the commands by updating member IKJTSO00 (for TSO Version 1.4 and above) of SYS1.PARMLIB, otherwise install the TSO command authorization SMP/E USERMOD provided in the base installation.
2. In TSO/E (TSO Extensions) Version 2 environments the CA-ASM2 ISPF Dialog Manager invokes all CA-ASM2 commands using the TSOEXEC service. Thus, commands requiring authorization can execute properly.

2.4.1.5 MVS APF List

The library containing CA-ASM2 load modules must be APF authorized. To authorize a load library, the library must be one of the following:

- A linklist library (if LNKAUTH=LNKLST in IEASYSxx defaults linklist libraries to being authorized)

  or

- A library with its data set name and volume serial number specified in the IEAAPFxx member of SYS1.PARMLIB.

Most CA-ASM2 batch functions must be executed from an APF-authorized library. To do this, establish CAI.CAILIB (or whatever you have called it) as an authorized library by adding its name to the appropriate IEAAPFxx member of SYS1.PARMLIB.

2.4.1.6 MVS Linklist Option

The target load module library for CA-ASM2 is CAI.CAILIB (or whatever high-level qualifier you may have chosen to replace CAI). This data set may be specified as a member of the MVS linklist library concatenation, if desired. The appropriate LNKLSTxx member of SYS1.PARMLIB needs to be updated if you choose this option. If you do not choose to add CA-ASM2 to the MVS linklist, you must add STEPLIB DDs for CAI.CAILIB to the CA-ASM2 procedures and the TSO logon procedures of those TSO users who use the CA-ASM2 TSO commands. During installation, it is best to utilize a STEPLIB to test out the new version of the product, even if you plan to add CAI.CAILIB to the MVS link list.

2.4.2 Aging

CA-ASM2 operation depends on the availability of data set activity information stored in the VTOC and/or VSAM (or ICF) catalog. This information can be maintained by CA-ASM2 programming, client programming, or mainstream IBM operating-system programming. The CA-ASM2 facilities for archival, incremental backup, and usage reporting can process the activity information regardless of its format and how it is stored.

Your installation may decide to use existing in-house routines to maintain data set activity information because the routines predate CA-ASM2 and other functions depend on them. For example, the routines may also maintain security or accounting data in the VTOC records.

The main advantage to using the IBM facilities for recording data set activity is that no system modifications are required. Through MVS Selectable Unit 60 (SU60), IBM permits recording the date of last reference to, and changed/unchanged status for, non-VSAM data sets. With the DFP product, the same information can be maintained for VSAM clusters cataloged in ICF catalogs.

There are several advantages to using the CA-ASM2 data set activity recording in addition to IBM's. First, more information can be recorded. For non-VSAM data sets, CA-ASM2 can maintain:

- Cumulative use count
- Last modification date
- Last modification time
- Identifier of either the first or latest update within one day. (This can be the job name or user ID, JOB card accounting data, or CA-ACF2 logon ID.)

Second, CA-ASM2 offers a quiet OPEN facility that prevents update of last reference information for OPENs from specified programs. This is useful in preventing utilities that can open data sets (such as MAPDISK) from updating the last reference date.

2.4.2.1 Aged DSCB Formats

In the VTOC, the format of an aged DSCB differs from a non-aged DSCB in the contents of the fields listed next. This information is placed into the DSCB by OPEN, so data sets that are not opened do not have this information.

Item	Description	Offset	Format
USECNT	Usage Count (#OPENS)	45	F
LMDATE	Last Modification Date	48	XL3, binary YDD
LMID	1st/Last Modifier ID	62	CL8
SU60	Change/Unchanged Flag	70	X1
LMTIM	Last Modification Time	71	XL2, HHMM
LSTUS	Last Usage Date	75	XL3, binary YDD

The offsets are in decimal, include the 44-byte dsname key and are relative to 1 (not 0).

Prior to DFP 3.1 LMTIM was stored at offset 80. See "Considerations for Current CA-ASM2 Clients" on page 2-29 if you have not applied this modification.

The $CNVAGE exit converts the format of the DSCB from that which is on disk to a format which is expected by the CA-ASM2 routines. The $CNVAGE exit provided with the system accepts several standard CA-ASM2 DSCB formats, SU60 format, CA-3 format, and an SU60-compatible format. It converts the data available into an internal standard format. $CNVAGE can also be used to reformat data kept by customer programmed open modifications.

The SU60-compatible format simply interchanges the usage count and last usage date fields from the above format. The usage count is kept in three bytes and is limited to X'45FFFF' or about 4.5 million. This change allows the last usage date to be in the same location as used by SU60. The last usage date is at offset 75 (decimal), and the usage count is at offset 45 (decimal) in the DSCB in the VTOC.

Note: Previous users of this modification should use this new version of the OPEN modification when implementing Version 4.2. If you are a current client and have not applied maintenance to install DFP 2.4/SMS compatibility support, you need to perform the aging conversion described in Appendix F after you have installed Version 4.2 of CA-ASM2.

2.4.3 Operating System Requirements

CA-ASM2 operates under the MVS operating systems with the Base Control Program (BCP) at Level 3.8 or above. It is fully compatible with all MVS/SP, MVS/XA, and MVS/ESA systems.

CA-ASM2 operates under all versions of JES2 and JES3.

2.4.4 SMS Requirements

CA-ASM2 acquires the current SMS CLASSes during a reload request by pre-driving the ACS routines. To accomplish this, the X'80' bit of the $SOFTWAR option must be turned on, IBM PTF UW27044 must be applied, and the ACS routines altered as follows:

1. The first check in the ACS routine must check the &ACSENVIR variable for RECALL. If RECALL is found, the ACS routines set the SMS CLASSes. This variable is set to RECALL by CA-ASM2 any time a reload request is being processed.
2. If the above check does not find RECALL in the &ACSENVIR variable, the &PGM variable is checked for either $IXRMAIN, M2BMON, or $ASMBMON. If any of these program names are found in the &PGM variable, the ACS routines does not set the SMS CLASSes because they have already been set when CA-ASM2 pre-drove the ACS routines earlier in the reload process.

The other SMS ACS read-only variables that are specifically set by CA-ASM2 when it is pre-driving the SMS ACS routines are:

ACS Variable	**IXR Reload**	**Batch Reload**
&DSN	Original DSN	Original DSN or NEWNAME/NEWQUAL
&DSNTYPE (set for PDSE data sets only)	LIBRARY	LIBRARY
&DSORG	PS, PO, VS or DA	PS, PO, VS or DA
&RECORG (set for VSAM data sets only)	KS, ES, RR or LS	KS, ES, RR or LS

2.5 Storage Requirements

CA-ASM2 supports the following Count, Key and Data mode DASD devices:

3340
3350
3375
3380
3390
9345

The CA-ASM2 data sets and support files are described in detail in Appendix A of the *CA-ASM2 Getting Started* guide. There are three sets of files to consider:

Distribution files for CA-ASM2
Permanent files for CA-ASM2
Permanent target files for Computer Associates products

2.5.1 Distribution Files for CA-ASM2

The following table shows the CA-ASM2 distribution files (DLIBs) and their approximate sizes.

Library Name	Blksize	Alloc Blks	Alloc Dir	Alloc Trks (3380)	Alloc Trks (3390)	Library Description
CAI.L142.CL142LLD	6144	1000	120	146	128	CA-ASM2 and IXR Load Library
CAI.L142.CL142MLD	3120	1900	150	152	133	CA-ASM2 and IXR Macro Library
CAI.L142.CL142SLD	3120	500	90	42	37	CA-ASM2 and IXR Source Library
CAI.AM42.CAM42LLD	6144	75	45	13	12	Allocation Manager Load Library
CAI.AM42.CAM42MLD	3120	60	15	6	5	Allocation Manager Macro Library
CAI.AM42.CAM42SLD	3120	80	15	8	7	Allocation Manager Source Library
CAI.NS10.CNS10LLD	6144	110	10	15	13	CA-RSVP Base Load Library
CAI.NS10.CNS10MLD	3120	375	25	26	22	CA-RSVP Base Macro Library
CAI.NS10.CNS10SLD	3120	120	5	9	8	CA-RSVP Base Source Library
CAI.NU42.CNU1042LLD	6144	150	20	20	17	CA-ASM2 Common Base Load Library
CAI.NU42.CNU1042MLD	3120	40	10	4	3	CA-ASM2 Common Base Macro Library
CAI.NU42.CNU1042SLD	3120	10	2	3	3	CA-ASM2 Common Base Source Library
Totals				381	333	

2.5.2 Additional Non-SMP/E Distribution Files

Library Name	Blksize	Alloc Blks	Alloc Dir	Alloc Trks (3380)	Alloc Trks (3390)	Library Description
CAI.NS10.TABLE	5164	10	0	1	1	Sample CA-RSVP Commands table
CAI.L142.PARMLIB	3120	120	20	11	9	Sample PARMLIB
CAI.NU42.SAMPJCL	3120	500	90	42	37	Sample JCL
Totals			110	54		

2.5.3 Permanent Files for CA-ASM2

The permanent files for CA-ASM2 are described in Appendix A of the *CA-ASM2 Getting Started* guide. The default sizes amount to approximately 700 tracks of 3380 or 3390 space. The default sizes should be adequate for most installations.

If you are uncertain as to your space requirements, use the defaults for installation. Later, you can adjust your allocations as needed.

2.5.4 SMP/E Installation Target Files

The target files for Computer Associates products are permanent files which may have been created during the installation of this or other Computer Associates products. These files may be used by several components or products. If these files were allocated with another product installation, you must ensure that enough space is available in them for CA-ASM2, however, the default allocations for installation include only enough space for CA-ASM2. The table on 2-14 shows the permanent files used by Computer Associates products.

If you plan to install CAIRIM from the CA Common Services tape prior to installing CA-ASM2 and would like to install both products into the same set of libraries, ensure that the permanent target files allocated during the CA Common Services installation are large enough to also hold CA-ASM2.

If you have already installed CAIRIM from the CA Common Services tape, some, but not all, of the permanent target files listed next have already been allocated. Review the following data set space requirements against the available space in the data sets allocated by the CA Common Services components installation. You must modify the job L140IA1 executed in Step 3 to allocate only the data sets not previously allocated by the CA Common Services components installation. No modifications are necessary if a separate set of data sets are allocated for CA-ASM2 exclusively.

Library Name	Blksize	Alloc Blks	Alloc Dir	Alloc Trks (3380)	Alloc Trks (3390)	Library Description
CAI.CAICICS	6144	6	5	2	2	Common CICS
CAI.CAIISPL	6144	6	5	2	2	Common ISPF Load
CAI.CAIISPP	3120	300	75	26	22	Common ISPF Panel
CAI.CAIISPM	3120	450	90	38	32	Common ISPF Msg
CAI.CAIISPS	3120	12	5	3	3	Common ISPF Skeleton
CAI.CAIISPT	3120	12	5	3	3	Common ISPF Tbl
CAI.CAILIB	6144	1020	100	149	131	Common Load
CAI.CAILPA	6144	50	45	9	8	Common LPA
CAI.CAIMAC	3120	700	90	56	49	Common Macro
CAI.CAIPROC	3120	100	90	10	9	Common Procedures
CAI.CAISRC	3120	700	120	57	50	Common Source
CAI.CAICLIB	3120	600	30	47	41	Common CLIST
CAI.PPOPTION	3120	200	90	18	16	Common Options
Total				420	368	

Note: The Computer Associates common files CAI.CAICICS, CAI.CAIISPL, and CAI.CAILPA are not used by CA-ASM2 and are only given a minimal allocation.

The Common files actually used for CA-ASM2 and its associated components are:

CAI.CAIISPP
CAI.CAIISPM
CAI.CAILIB
CAI.CAIMAC
CAI.CAIPROC
CAI.CAIISPT
CAI.CAISRC
CAI.PPOPTION
CAI.CAICLIB

2.6 CAIRIM Requirements

CAIRIM is used to install CA-ASM2, IXR and Allocation Manager system modifications. CAIRIM is the common driver for a collection of dynamic initialization routines that eliminate the need for preinstalled user SVCs, SMF exits, subsystems, and other installation requirements commonly encountered when installing systems software. These routines are grouped under the Computer Associates MVS Dynamic Initialization component S910.

CAIRIM operates under the MVS operating systems with the Base Control Program (BCP) at Level 3.8 or above. It is fully compatible with all MVS/SP, MVS/XA, and MVS/ESA systems.

CAIRIM must be installed into an APF authorized library. Linklist is not required. However, it is recommended to help prevent errors caused by modules not being available. CAIRIM is installed into the common CAI.CAILIB.

CA-ASM2 uses the CAISMFI Dynamic SMF Event Interceptor, so SMF recording must be active in the system for any SMF Event or Data handling product routines. CAIRIM's SMF Interceptor component does not require any particular SMF records. In some cases, a particular Computer Associates product requires certain SMF records be recorded, CA-ASM2 does not.

If your installation already has CAIRIM installed, it must be at the 9203 level. This is required for the subsystem initialization module support required by Allocation Manager. If your level of CAIRIM is not current enough, you must apply maintenance to bring CAIRIM up to the current level. A CA Common Services Maintenance tape may be included in the materials shipped with CA-ASM2 Version 4.2, otherwise it should be ordered from Computer Associates.

CAIRIM requires approximately 12K of ECSA and 4K of CSA for MVS/XA or MVS/ESA systems or 16K of CSA for non-XA systems. Additional requirements for resident modules vary with each component and product. The CA-ASM2 IXR component uses 12K of ECSA and 64K of CSA for MVS/XA or MVS/ESA systems or 72K of CSA for non-XA systems for programs and control tables, approximately 8K of ECSA on XA/ESA or CSA on non-XA for IXR control blocks and 10K for IXR system modifications if these are installed with CAIRIM.

For additional information on CAIRIM, see the *CA Common Services Administrator Guide*.

2.7 License Management Program (LMP)

Beginning with Genlevel 9209, CA-ASM2 is maintained by the Computer Associates License Management Program (LMP). LMP provides a standardized, automated approach to license managment and is designed to eliminate the difficulties inherent in tracking the accurate, licensed use of Computer Associates products. LMP and an optional online facility comprise Total License Care (TLC), a component of Computer Associate's comprehensive service and support program known as CA-TCC: Total Client Care.

If you installed and maintain CA Common Services through SMP/E, please see Task 13a (Customize the CAIRIM procedure) in *CA Common Services Getting Started*. This task explains the modifications needed to the CAIRIM procedure to support LMP.

You should have received an **LMP PRODUCT KEY CERTIFICATE** which includes the CPU ID for each CPU at your site that is licensed to run CA-ASM2 at the current maintenance level. See Task12a in *CA Common Services Getting Started* for information on updating the new KEYS member in the CAI.PPOPTION data set.

If the CPU ID in the LMP KEY is the CPU ID of the machine on which you are running CA-ASM2, you do not see any warning message, and CA-ASM2 runs without any additional changes until the expiration date in the LMP KEY is reached. Approximately 30 days prior to the expiration date, a warning message starts to be issued. This warning message includes the name of the product that is about to expire and the number of days remaining before expiration.

If you do not have a valid LMP KEY for the CPU on which you are running CA-ASM2, CA-ASM2 issues a console message stating that the CPU on which you are running does not have a valid LMP KEY.

If, after referring to the documentation mentioned above you still have questions, please call the Computer Associates License Management Program Hotline at 1-800-338-6720 in North America (24 hours a day, 7 days a week). Outside of North America, contact your local Computer Associates Technical Support Center during local business hours.

2.7.1 LMP Key Certificate

Computer Associates LMP provides a standardized and automated approach to the tracking of licensed software. Computer Associates LMP definitions are required to initialize CA-ASM2 properly. Examine the Computer Associates LMP Key Certificate you received with your CA-ASM2 installation tape. This certificate contains the following information:

Item	Description
Product Name	The trademarked or registered name of the CA-ASM2 licensed for the designated site and CPUs.
Product Code	A two-character code that corresponds to CA-ASM2.
Supplement	The reference number of your license for the particular CA-ASM2, in the format nnnnnn - nnn. This format differs slightly inside and outside North America, and in some cases may not be provided at all.
CPU ID	The code that identifies the specific CPU for which installation of your CA-ASM2 is valid.
Execution Key	An encrypted code required by CA LMP for Computer Associates-product initialization. During installation, it is referred to as the LMP Code.
Expiration Date	The date (ddmmmyy, as in 15JAN93) your license for CA-ASM2 expires.
Technical Contact	The name of the technical contact at your site who is responsible for the installation and maintenance of the designated CA-ASM2. This is the person to whom Computer Associates addresses all CA LMP correspondence.
MIS Director	The name of the Director of MIS, or the person who performs that function at your site. If the title, but no individual name, is indicated on the certificate, you should supply the actual name when correcting and verifying the certificate.
CPU Location	The address of the building where the CPU is installed.

2.8 Multi-CPU Installation Considerations

CA-ASM2 is designed to function in a single CPU or in a multi-CPU environment where DASD is shared between systems. The following issues should be considered when installing CA-ASM2 in such an environment:

- Put CAI.CAILIB on a shared DASD volume accessible to all systems.
- Put the IPC and Journal data sets on (separate) shared DASD volumes accessible to all systems.
- Authorize CAI.CAILIB on all systems needing CA-ASM2 functionality.
- For each CPU ID that you have, you must have a separate LMP PRODUCT KEY certificate.
- During installation, any task which generates an operating system modification such as the CA-ASM2 Aging Modification to OPEN module IFG0196W, may need to be reexecuted with parameters provided for the other target operating system(s). This is required only if the full CA-ASM2 functionality is required on the other systems. For example, in some VM environments, one machine is the production system on which full CA-ASM2 functionality is required and additional machines are IPLed for testing purposes. You should evaluate which components of CA-ASM2 are needed on the test systems and install the associated operating system modifications if needed.
- If the additional operating system is at the same level as the base operating system CA-ASM2 was installed on, the modules may be copied from the base system to the other operating system. During any such module copy operations, ensure that all aliases are copied with the modules.
- To install the system modifications on additional operating systems at a different level from the base system, you must define a new data set to contain the USERMODs. This data set must be used in place of CAI.CAILIB when generating the system modifications, otherwise you replace those generated for the base system. All system modifications are created in CAI.CAILIB, whether CAIRIM or SMP/E USERMODs are used to install them. To override CAI.CAILIB and use the new data set you must edit the generated task JCL to specify your new data set on the CAILIB DD statement following the MVS SMP/E procedure (do not edit the ++JCLIN Link DD statements). Be sure to change it back to CAI.CAILIB when the SYSMODs have been generated successfully.
- The SMF system ID may optionally be kept as part of the IPC record key. This may be used to distinguish data sets unloaded on one system from data sets with the same name which are unloaded from another system. IXR is the main CA-ASM2 component which uses the system ID in its processing. If the SMF system ID of the system IXR is running on is present in an IPC record key, IXR reloads the data set version which matches the SMF system ID of the system IXR is running on. This causes IXR to bypass a more current version of the same data set if it is known to be associated with another MVS system.

If your environment requires that IXR be able to associate data sets with a given MVS system in a multi-CPU environment, you should include the SMF system ID in the key of the IPC records which are tied to a specific system, otherwise you should leave it out. The $CATMRG SYSID=Y parameter of $DASDMNT is used to request that the SMF system ID be included in the IPC key at unload time. An example of an environment where $CATMRG SYSID=Y should be specified on selected archive jobs follows:

1. Two MVS systems are running with shared DASD (MVSA and MVSB).
2. A single CA-ASM2 IPC is shared between these two systems.
3. CA-ASM2 archive and/or backup procedures are run from both of the systems.
4. IXR is running on both of the systems.

In this environment, you should separate archive processing into two groups, those archive jobs which request that the SMF system ID be included in the IPC record key and those which do not.

Most archive and backup processing should be done against the shared DASD volumes, which contain data sets cataloged in user catalogs shared between the systems. These jobs should request that the SMF record ID NOT be kept in the IPC key ($CATMRG SYSID=N).

In addition, an archive job should be set up for each MVS system to archive the data sets associated only with that system. For these jobs $CATMRG SYSID=Y should be specified to keep the SMF system ID in the IPC key. The best way to manage this scenario is to use CA-RSVP as a preprocessor to explicit archive. To identify which data sets are tied to a specific MVS system, the catalog structure must be investigated. Data sets which are cataloged only in the system's master catalog or which are cataloged in a nonshared user catalog should be processed using the $CATMRG SYSID=Y option. Use CA-RSVP to specify the high-level qualifiers of these data sets and to also specify the criteria for archival (not used in 90 days, and so on).

SYS1.AMODGEN is an example of a data set which should be processed with an archive job specifying $CATID SYSID=Y. A copy of this data set resides on both MVS systems and it is defined only in the master catalog of each system. (This data set is selected for illustrative purposes only, CA-ASM2 never archives "SYS1" data sets.) If the SMF system ID is not retained in the IPC key, IXR would not be able to distinguish a version of SYS1.AMODGEN archived on MVSA from a version archived on MVSB, and would simply reload the most current one. This could result in the wrong version being reloaded for the MVS system requesting the reload.

See the $CATMRG SYSID=x parameter of $DASDMNT and the SYSID=x JCL parameter of the CA-ASM2 catalog conversion program M2PCCNV for further information regarding SMF system ID processing.

2.9 Disk Staging Area Planning

The DSA is a disk pool or disk SMS storage group that holds user data sets which CA-ASM2 has archived or backed up. Archivals and backups may be made directly to tape, or you have the option of selecting the Disk Staging Area (DSA) as the output area for any backup or archive run. If the DSA is used rather than tape, CA-ASM2 can respond to reload requests more quickly without requiring operator intervention.

The DSA is not intended to be used instead of tape; but rather to be used in conjunction with tape. When used thoughtfully, it makes the entire archive, backup, and reload process run more smoothly and quickly. It enables the archive or backup process to be run without an operator being on duty. By avoiding tape mounts and the accompanying rewind and unload times, the archive or backup process completes in less elapsed time.

CA-ASM2 is designed to run with two Disk Staging Areas; one for archive and one for backup. Each DSA has a limited capacity. To gain the maximum advantage from using a DSA, its space is used conservatively. When CA-ASM2 writes into the DSA, it compresses and reblocks the data to make full use of the track capacity. CA-ASM2 also offers several mechanisms to select which data sets are to be written into the DSA and for how long they are to stay. Selection criteria are supplied to $DASDMNT to choose which data sets are to be written into the DSA. $DASDMNT assigns a DSA retention period to each data set. For SMS controlled data sets, $DASDMNT uses the management class field "Level 1 Days Non-Usage" to set the retention period. For any data set not having a management class, $DASDMNT uses the $DSADAYS parameter.

Any data set written to the DSA should be copied to tape at the next opportunity. CA-ASM2 keeps track of both the DSA copy and the tape copy. For reloading purposes, CA-ASM2 ***always*** considers the DSA copy as the primary copy of the data set. Once a data set is copied from the DSA, it becomes a candidate to be purged from the DSA, but may remain in the DSA for quite some time, depending on the options chosen.

Note: Volumes having a volume attribute of PRIVATE may not be used for a DSA.

2.9.1 Managing the Disk Storage Area

The DSA Manager (M2DSAMGR) is the CA-ASM2 utility that manages the data sets within the DSA. This utility is able to:

- Copy data sets from the DSA to CA-ASM2 tapes (COPY and MOVE commands).
- Purge data sets from the DSA (PURGE command).
- List the contents of the DSA (LISTDSA command).

This utility must be run periodically to copy the DSA data sets to tape. Although you may choose to run the DSA manager to purge expired data sets from the DSA to free up space in the DSA, CA-ASM2 automatically reclaims DSA spaces as it needs it.

There are six different ways a data set in the DSA may be purged:

1. The data set expires and is removed by IPC maintenance ($MAINT).
2. An explicit delete from archive or backup command ($DA or $DB) is issued.
3. M2DSAMGR is run with a MOVE command.
4. M2DSAMGR is run with a PURGE command.
5. The DSA becomes full while $DASDMNT is writing to it, so $DASDMNT automatically purges eligible DSA data sets.
6. The data set is reloaded and a tape copy also exists.

If the archived data set expires or if an explicit delete from archive ($DA) command is issued for a given archive version, that version is removed from the Integrated Product Catalog (IPC). Any copy of that data set residing in the DSA is deleted.

If a PURGE command is issued, all archived data sets fitting the criteria is removed. However, the PURGE command is quite different from the $DA command in that PURGE ***requires*** there to be a tape copy present in the system. The PURGE command effectively switches the primary version from disk to tape by deleting the disk version and making the tape copy the primary copy.

Automatic purging only occurs when the DSA becomes full. Automatic purging is quite similar to the M2DSAMGR PURGE command, but is designed to function or complete without operator intervention, so automatic purging may remove data sets from the DSA before their appointed time. The main archive and backup utility, $DASDMNT, whether running as scheduled batch or performing threshold archival ***always*** uses automatic purging of the DSA to make room for newly archived or backed up data sets. Automatic purging ***never*** removes a DSA data set unless a tape copy exists for that data set.

When the DSA becomes full, $DASDMNT initiates automatic purging by invoking M2DSAMGR to purge data sets from the DSA. M2DSAMGR analyzes all of the data sets in the DSA by scanning through the IPC and identifying all IPC records whose Unload Destination Cell (UDC) contains a DSA Destination SubCell (DDSC). Only data sets which have already been copied to tape are candidates for this automatic deletion. M2DSAMGR calculates the number of days left before each data set is to be removed from the DSA and then sorts them according to when they are scheduled to be removed and by their size. Data sets that have already been in the DSA for their

allotted time are removed first. If no data sets of this description remain and the DSA is still full, data sets that are nearest to reaching their allotted time are selected for removal, but these are reported as exceptions so the data administrator can increase the size of the DSA. This technique allows for the greatest number of data sets to be in the DSA while still providing the flexibility of removing as many of them as needed without operator intervention.

When any DSA data set that has already been copied to tape is reloaded back to primary storage, the DSA copy of the data set is purged. Any future reloading of this archive or backup version is made from the tape copy.

2.9.1.1 Data Sets in the DSA

Each data set archived or backed up to the DSA is written in a type of VBS format. These data sets are RECFM=VBS, except that the LRECL can reach 64K instead of the DFP limitation of 32760 bytes. In these data sets, each logical record is a data segment written by the unload data mover. It is these same segments that are expected by the reload data mover when these data sets are reloaded. For the logical data movers, each logical record in the DSA represents a full block of data. For the physical data mover, each logical record represents a track image. The Disk Staging Area Format Manager, M2DSAFMT, processes each logical record by compressing it and then adding it to the output DSA data set in VBS fashion. The physical block size chosen for the DSA data set is optimum for the disk device where it is recorded and is totally unrelated to the original data set's BLKSIZE.

2.9.2 DSA Data Set Name Format

The $DPREFIX field in $OPTIONS defines the high-level qualifier(s) of data sets written into the disk staging area. Like the other prefix fields, this field is limited to 21 characters. Every data set archived or backed up to disk begins with this qualifier.

This field is concatenated with a time stamp field in the format **uyyyyddd.Thhmmss.annnn** where:

u is either A for archived data sets or B for back up data sets.

yyyyddd is the Julian date the archive or backup run started.

T is always T.

hhmmss is the time of day the archive or backup run started. The date and time used in generating the name matches the unload date and unload time stored in the unload record in the IPC, so every data set in a single run has the same unload date and time.

a is an alphabetic character from A to Z. The first 9999 unloaded data sets have **a** = A, the second 9999 unloaded data sets have **a** = B, and so on.

nnnn is an ascending number from 0001 thru 9999.

2.9.2.1 Empty Data Sets

The way CA-ASM2 handles an empty data set depends on the DSORG of that data set. When an empty sequential data set is archived or backed up to tape, CA-ASM2 writes out a set of header and trailer records to indicate an empty data set on tape. For these empty data sets in the DSA, CA-ASM2 allocates a data set of zero tracks in length. This uses a DSCB, but does not occupy any space for the data itself. Empty data sets of all other DSORGs have control information written into the data set in the staging area.

2.9.2.2 Backup Considerations

The DSA provides the ability to back up data sets to disk rather than to tape. The disk backup volumes can be viewed as a backup spool or transient area where backed up data sets are held until they can be copied to tape volumes. Data sets that are backed up to disk are compressed and reblocked to conserve disk space. This technique can substantially reduce the backup window by eliminating the need for tape device allocations and tape mounts. It also allows you to perform backups without operator intervention. You may continue to use disk-to-tape backup in some circumstances while using disk-to-disk backup in others.

On a regular basis, you should run M2DSAMGR to copy the data sets in the backup DSA to tape. When data sets are copied to tape, you have the option of decompressing and blocking the data with the SYSIN input to the M2DSAMGR program. This results in the data being decompressed and blocked according to the data mover that must reload them from tape. This provides maximum recoverability in the event the client does not have CA-ASM2 Version 4.2 available at a recovery site and it also eliminates the need to develop a number of independent reload utilities. M2DSAMGR removes backed up copies from the DSA when they expire or when they are copied to tape with the MOVE command.

2.9.2.3 Runtime Considerations

In addition to regularly scheduled archival and backup runs, automatic threshold archival may cause unscheduled CA-ASM2 activity. These tasks require access to the CA-ASM2 IPC and require some system resources. This should only occur when a space related abend and subsequent rerun would cost even more in lost time and system resources. If IPC access contention between IXR and M2DSAMGR becomes a problem, you should adjust the IPC-ENQTIME(x) and IPC-FREETIME(x) parameters to set up shorter access windows. All of the parameters are described in the *CA-ASM2 System Reference Guide*.

2.9.3 Security Considerations

The high-level qualifier used to identify data sets in the DSA must be secured. Only CA-ASM2 functions relating to archival, backup, IPC maintenance, and DSA maintenance should be authorized to create or destroy data sets in the DSA. It is recommended that you secure the DSA volumes themselves so no other data sets can be allocated there. Only the CA-ASM2 and $DASDMNT reloads, and DSA maintenance functions should have read access to data sets in the DSA. In addition, the M2DSADEL program should be authorized separately to perform a delete from the archive function.

2.9.4 DSA Archival

Archival is probably the key area for DSA use, mainly because of IXR and the removing of requirements for tape mounts. Also, there is a lot less archival activity going on than backup and you do not have to use the DSA for all of your archive since in a given $DASDMNT run, you specify whether you want to go disk-to-disk.

If you are an existing CA-ASM2 user, you can use CA-RSVP to do tentative selection like simulating what your archival run might do and how much DSA space would be required. For instance, you could use CA-RSVP and select all the data sets that have not been used in the last 90 days or so and get a feel for the volume of data.

During an archive run the actual compression of data itself is dependent on the data content. In the case of VSAM, which is typically allocated in cylinders with embedded free space, you can expect a higher percent overall compression realized. CA-RSVP can be used to run against your volumes to make a tentative selection total by looking at the number of tracks and reducing that figure by the compression factor. The result is close to the amount of DSA required for the archival.

The assumption is that when you put data into the DSA, at some point in time like every Saturday afternoon, you run one of the DSA COPY functions that makes a copy of what is in the DSA on tape. Normally you want everything that is in the DSA to have a tape copy as well. You may leave data in the DSA as long as possible, but at some point in time the DSA becomes full. Whenever the DSA becomes full the DSA manager is invoked during the $DASDMNT unload run. A purge request is given to find enough room and, to find enough room, it searches through all the data sets in the DSA that have a tape copy. Those that have a tape copy are the eligible ones to be purged. $DASDMNT then looks for the ones that are closest to meeting their expiration date and it starts purging those until it frees up enough room. It is usually targeting the threshold and it uses the same default thresholds that are specified in $OPTIONS. It then purges the DSA until it is down as close to this parameter as it can and $DASDMNT resumes archiving into the DSA.

When a data set is put into the DSA it is assigned its own DSA expiration date which is different than the expiration date of the data set itself. You could archive a data set and keep it for five years, but only keep it in the DSA for 120 days. So the DSA expiration date is computed as 120 days after the day it is first put in the DSA. Each

data set therefore has two expiration dates, the DSA copy's number of days to keep it and the actual days until it is purged.

For SMS environments, the management class has a field called LEVEL1 days which is used instead of the $DSADAYS parameter. If you are running through many SMS controlled data sets, each data set has its own unique DSA expiration date assigned to it. In a given run, you could have one data set that is to be kept in the DSA for 60 days, another to be kept in the DSA for 180 days, and another to be kept for 30 days. When looking for a candidate for purging, the one that is closest to its $DSADAYS or LEVEL1 days, that is, closest to meeting its expiration date is looked for. You might have a data set in the DSA that has been there for 120 days but it has another 80 days to go before it is destined to be purged. There may be another data set that has been in there for two weeks, but it has only a three week retention period, so it is much closer to meeting its expiration date and is picked over the one that has been in there for 120 days. It is possible in a DSA that is very large you could have a data set that is designated to be kept for 60 days in the DSA, but if the DSA is empty, it is kept for 90 or 120 days or as long as possible. For instance, if the data set is specified to be retained for two years, and you have a large DSA, it may never be purged, and may stay in the DSA for its entire life. When the data set finally expires at the end of the two years $MAINT then purges the DSA data set or copy.

The minimum space recommendation for your archive area in the DSA is one volume, setting up either an SMS storage group or esoteric unit name with a name like ARCHIVE so that you can specify UNIT=ARCHIVE or have a storage group called ARCHIVE and an associated storage class may be called ARCHIVE which triggers the ACS routine to select that storage group. Computer Associates recommends nothing else but disk-to-disk archive on DSA volumes.

If you have Version 4.0 already installed you can run a listing using CA-RSVP to summarize the amount of tracks represented in the archive records in their IPC. For instance, you could run CA-RSVP against the IPC and specify to list out all the data sets that have been archived in the last 120 days to determine the number of tracks used. Given that number of tracks you can cut it by 50 to 70 percent to arrive at an estimated number of DSA tracks you would need to store those archive records.

A lot of times VSAM data is highly compressable because when it is archived the index requires no space. If you have a KSDS that has an embedded index a track of every cylinder is saved. There is no embedded free space and there is no problem with spanning. So a number of things work together on top of the data compression to greatly reduce the amount of space required. In some cases it could reach 90 percent, so if you estimate 50 to 70 percent compression you are generally making a conservative estimate.

2.9.5 DSA Backup

If you look at the total bulk of data that is moved, backup is the most massive amount of data moved, because you do it over and over. Disk-to-disk backup is used for different reasons than disk-to-disk archive. One of your main objectives may be to reduce the backup window. There are several things that have been put in to facilitate disk-to-disk backup. One of them is no requirement for any LOXXX data sets. In the past to have multiple $DASDMNTs running, for each $DASDMNT you were going to run you had to have a separate LOXXX data set, a separate INCR file and IDs that would help distinguish each of these runs and its collection of data sets that were required to run that particular copy of $DASDMNT. For disk-to-disk backup, none of these is required.

A new feature has been added specifically for disk-to-disk backup to allow concurrency. This new facility for disk-to-disk backup allows you to have just one set of control statements that defines everything in the data center to be backed up, and you can submit as many copies of $DASDMNT as you want, all pointing to that same set of control statements. The various $DASDMNTs running in the system and their various address spaces coordinate among themselves. The first $DASDMNT in takes the first selection. The next $DASDMNT then would come in and discover that the first $DASDMNT already has the first volume, so it would then start working on the second volume. The next $DASDMNT would take the next available slot and so forth. It is like there was one list and you cascade down through the list. As soon as one of them finishes with the volume it takes the next available volume on the list that no other copy of $DASDMNT has already processed.

You could run four or five $DASDMNT copies or as many as you wanted to run to reduce the overall backup number by getting as much parallel processing as required. There is no tape mounts or rewind time, and you are not limited by the number of tape drives. The idea is to just keep sending all the data to the DSA backup. If you want you can run multiple backup areas, for instance, if you thought you would run four or five $DASDMNTs, you would have the option of setting up a DSA for each backup to minimize head contention. This allows you to run a shorter backup window and, better yet, an operatorless backup window. It helps facilitate an automatic operation where you can queue up to run at 2:00 in the morning and the whole backup process can take place at that time. In the morning during full production you can run the DSA manager with the MOVE command that moves the backup copies from the DSA onto tape, thus freeing up the DSA. You would do that for different reasons than you would for the archive. The main advantages are shortening the backup window, and allowing you to do processing after-hours without an operator present to mount tapes, and be able to copy it off to tape the next day.

2.10 Installation Materials

The following materials are provided for the installation and use of CA-ASM2:

- Planning Guide
- Getting Started
- System Reference Guide
- ISPF User Interface Guide
- Message Guide
- *CA-RSVP User Guide*
- *CA Common Services Getting Started*
- *CA Common Services Administrator Guide*
- A standard label 3480 cartridge or 3420 reel recorded at 6250 BPI, containing CA-ASM2 and all related components.

The following chart shows the layout of the tape:

File #	File DSN	File Description
1-8	Reserved	
9	CAI.SAMPJCL	CA-ASM2 Sample JCL library
11	CAI.L142.PARMLIB	CA-ASM2 PARMLIB
12	CAI.NS10.TABLE	CA-RSVP table
14	CAI.BKMGR	Contains Bookmanager documentation
28	CAI.PPOPTION	PPOPTION file
32	SMPMCS	SMP/E Control Statements
33	CL142B0.F1	CL142B0.F1 Base CA-ASM2 Macros library
34	CL142B0.F2	CL142B0.F2 Base CA-ASM2 Source library
35	CL142B0.F3	CL142B0.F3 Base CA-ASM2 Load library
36	CL142B0.F4	CL142B0.F4 Base CA-ASM2 ISPF Panel library
37	CL142B0.F5	CL142B0.F5 Base ISPF Message library
38	CL142I0.F1	CA-ASM2 IXR Macro library
39	CL142I0.F2	CA-ASM2 IXR Source library
40	CA142I0.F3	CA-ASM2 IXR Load library
41	CL142A0.F1	CA-ASM2 ISPF ????????
42	CL142A0.F2	CA-ASM2 ISPF ????????

File #	File DSN	File Description
43	CL142A0.F3	CA-ASM2 ISPF ????????
44	CL142A0.F4	CA-ASM2 ISPF ????????
45	CL142A0.F5	CA-ASM2 ISPF ????????
46	CL142A0.F6	CA-ASM2 ISPF ????????
47	CL142A0.F7	CA-ASM2 ISPF ????????
48	CAM42X0.F1	Alloc. Mgr XA/ESA Macros library
49	CAM42X0.F2	Alloc. Mgr XA/ESA Source library
50	CAM42X0.F3	Alloc. Mgr XA/ESA Load library
51	CNS1000.F1	CA-RSVP Macros library
52	CNS1000.F2	CA-RSVP Source library
53	CNS1000.F3	CA-RSVP Load library
54	CNS1000.F4	CA-RSVP ISPF Panel library
55	CNS1000.F5	CA-RSVP ISPF Message library
56	CNU4200.F1	CA-ASM2 Common Macro library
57	CNU4200.F2	CA-ASM2 Common Source library
58	CNU4200.F3	CA-ASM2 Common Load library
59	CNU4200.F4	CA-ASM2 ISPF Panel library
60	CNU4200.F5	CA-ASM2 ISPF Message library

Note: File 15 and files 30 through 32 are dummy files not used in the installation.

2.11 Considerations for Current CA-ASM2 Clients

If you are already a CA-ASM2 client and are installing Version 4.2 of CA-ASM2 on your production system, there are some important issues to consider.

Though your final objective is to replace your current production version of CA-ASM2 with the new version, Computer Associates recommends that you install the new version separately and switch it to production after you test the functions that are critical to you. This is particularly important if you have implemented any CA-ASM2 user exits in your current version. **Always reassemble and test these exits to confirm that they work correctly with the new version.**

There are several things to consider when installing a new version of CA-ASM2:

1. Version 3.1 and earlier of CA-ASM2 provided an optional step to define CA-ASM2 to an existing SMP/E environment. If CA-ASM2 was installed into a separate SMP/E zone, the old version may remain defined to SMP/E while installing and verifying Version 4.2. SMP/E also deletes the CA-ASM2 modules from the target data sets in performing the DELETE processing, unless you take action to prevent it. The ASM2DEL job provides an override to the ASM2MOD DD which should prevent any executable load modules from being deleted. If you modified the previous version SMP/E packaging to direct the modules to another target library, you should change the override to the target library DDNAME to which you directed them. It is a good idea to take full backups of the current "production" CA-ASM2 data sets prior to applying the dummy function to allow recovery in the event of a fallback.
2. Choose a new dsname prefix for the incoming distribution data sets and for the data sets created during the installation. Selected installation jobs delete the data sets they are about to create. If the same name as your current version is used, some production data sets could be scratched.
3. When defining the installation parameters, direct the CA-ASM2 JCL procedures to a library suitable for testing. Be careful not to use a library that allows the new procedures to be invoked by your current production CA-ASM2 jobs.
4. Keep the new CA-ASM2 load modules in a distinct library that is not a link list data set. They can be tested using a JOBLIB or STEPLIB DD statement. Foreground TSO command testing with a STEPLIB may be inconvenient. If so, test the commands in a batch job using the CA-ASM2 batch monitor (through the ASM2CMDU procedure) or the TSO batch TMP (IKJEFT01). Also, in TSO/E systems, you can use the TEST command with the CP parameter to test an entire library of authorized commands and related modules.
5. Be sure that user exits, $OPTIONS, and similar modules are correctly assembled for the new version and placed in the new version library. One of the most common problems in new version testing is fetching an old version exit or $OPTIONx module because it is missing from the test library and the production CA-ASM2 version is in the link list. Take the time to review the settings of all $OPTIONS fields and to ensure that the same settings are provided in the new $OPTIONS (except for DSN prefixes).

6. In general, the OPEN and CLOSE modifications, changes to the Program Properties table, IXR intercepts, I/O appendage routines, and similar system changes are upward-compatible from version to version. (If any are not, the installation bulletin for the new version will tell you.) You can test the new version of the base product and install it in production with the old system. If your installation is running an older version of CA-ASM2, it is a good idea to install the system modifications to ensure they contain current maintenance. See the planning sections for IXR and Allocation Manager for specific instructions on each component. See also the following item.

7. A new version of the OPEN modification IFG0196W was provided with Version 4.0 of CA-ASM2 and is again being distributed with Version 4.2. This OPEN modification saves CA-ASM2 aging information at new offsets in the Format-1 DSCB which are compatible with recent changes to the Format-1 DSCB required by SMS (DFP 2.4 and above). Current CA-ASM2 users with the IFG0196W OPEN modification installed (still using the old offset of 80 for LMTIM) are strongly encouraged to install the new version of IFG0196W due to the current maintenance levels of IBM products. Using the old offset for flag bits may cause unpredictable results. See Appendix E in the *CA-ASM2 Getting Started* guide for further information on VTOC cleanup required when installing the new version of OPEN modification IFG0196W.

8. Test the new version against your production Integrated Product Catalog (IPC) by making a $OPTIONx version of $OPTIONS for the new version that specifies the prefix used in your production CA-ASM2 data sets. Do this by modifying and reassembling the new version's $OPTIONS in CAI.L142.PARMLIB. If you are coming from a version prior to 4.2, do not copy $OPTIONS from the old version library to the new as there were several new $OPTIONS fields for 4.2. There were no new options added in Version 4.2. **Remember that if you give this new module the name $OPTIONS (rather than $OPTIONx, where x is not S),** all **tasks done by the new version access and update your production CA-ASM2 data sets by default.**

   Code `//$OPTIONx DD DUMMY` in all test job steps where `x` is the suffix you have chosen.

   Before executing functions in the new version against your production CA-ASM2 data sets, check the Product Information Packet (PIP) shipped with the installation tape to see whether the new version is compatible with those data sets. Some new versions change the content or structure of some CA-ASM2 data sets.

9. The Realtime Space Monitor data collection file (SMONFILE), had the logical record length expanded in Version 4.1. If a monitor file from a previous version is used by either JCL procedures ASM2SMON or ASM2SREP, message ASMRT815 is displayed and the programs terminate with RC=815. The proper allocation for the new monitor file is explained in the *CA-ASM2 System Reference Guide.*

10. Because of shared ARCLOG support in Version 4.2, you may want to combine multiple run IDs in concurrent runs. Use IEBGENER to create a merge for ARCH.ARCLOG and BKUP.ARCLOG.

2.11.1 Integrated Product Catalog (IPC)

The CA-ASM2 Integrated Product Catalog (IPC) contains information that previous to Version 4.0 was maintained in the archive, backup, and IXR catalogs. It also contains information about $DEFRAG full-volume backups which is used by the incremental recovery process.

The IPC is a VSAM key-sequenced data set containing variable-length records utilizing a cellular structure. An IPC cell is merely a collection of related data items. Separate cells have been defined for separate CA-ASM2 processes. The number and sequence of cells within an IPC record vary depending on the entity being tracked and its activity. The cellular structure employed in the IPC records allows CA-ASM2 to build records that contain only as much information as is needed.

The IPC integrity scheme is based on saving copies of the IPC at various points in time and on recording update activity in a journal. When CA-ASM2 has successfully backed up the IPC, it clears the Journal data set, and writes a special record into the Journal which holds the data set name and volume information of the backup tape(s) just created. The Journal is then available to record new activity until the next IPC backup cycle.

If the IPC is destroyed, the Journal contains all the information needed to complete the recovery. CA-ASM2 retrieves the backup data set name and volume information from the Journal. It then allocates the backup data set and copies its contents into the newly allocated IPC. Once the copy has completed, CA-ASM2 applies the update activity recorded in the Journal to the IPC to bring it up to date.

2.11.1.1 IPC and Journal Creation

The IPC and the Journal must both be allocated by any job or command which accesses the IPC. The names of these data sets and the ddnames by which they are known to CA-ASM2 is controlled by two $OPTIONS fields: $CATID and $VPREFIX. The ddnames are constructed from prefacing the one- to four-character CATID to IPC and JNL. For example if $CATID contains ASM2, the ddname for the IPC is ASM2IPC and the ddname for the Journal is ASM2JNL. If you specify these ddnames in your JCL, CA-ASM2 attempts to use the data sets they define as the IPC and Journal. If you omit them from your JCL, CA-ASM2 dynamically allocates them. The fully qualified data set name is constructed by concatenating the value contained in $VPREFIX and the related ddname.

For example, if $VPREFIX contains SYS2.PROD, the data set names for the ddnames mentioned above would be SYS2.PROD.ASM2IPC and SYS2.PROD.ASM2JNL, respectively.

Both of these data sets can be shared using VSAM SHAREOPTIONS(3 3) because CA-ASM2 controls the sharing of these data sets and assures the integrity of these data sets. The sharing technique uses RESERVE/RELEASE for protection. The $SOFTWAR flags in $OPTIONS can be used to change the RESERVE to an ENQ, but this is recommended only for testing purposes. The sharing technique also uses

the first physical record of each of these data sets as a control record to keep track of the internal structure of the data set. The integrity of the IPC and Journal can only be assured when CA-ASM2 components access the IPC. Other programs should not update the IPC or Journal outside of CA-ASM2's control, except where noted during the defining of the IPC and Journal.

Caution

For system integrity, the IPC and Journal should NOT reside on the same volume. Also, DFP Version 3.1 (SMS) users should not place the IPC or Journal on the same volume as the COMMDS or ACDS, due to the RESERVE activity against these data sets.

If you have a version prior to Version 4.0 installed and are installing Version 4.2, you need to convert to the new Integrated Product Catalog. The steps involved to convert your catalog to the new format for Version 4.2 are contained in Appendix ???? in the *CA-ASM2 Getting Started* guide. You should complete all steps of the installation for Version 4.2 before converting your catalog.

2.11.2 Aging Conversion

Since Version 4.2 ages in the new SMS-compatible format, aging conversion becomes a consideration for previous clients who have not installed the optional APAR tape A89299. Instructions for this aging conversion are contained in Appendix E in the *CA-ASM2 Getting Started* guide and should be performed after the installation of Version 4.2.

2.11.3 Fictitious Volsers

You can set up CA-ASM2 to recatalog Generation Data Sets (GDS) automatically to a fictitious volser. To do this, include the GDGONLY subparameter in the $RECATLG SYSIN parameter to the archive job. If you want, you can archive and recatalog GDSs separately, using the CA-ASM2 facility $GDGMON and the Explicit Archive procedure, ASM2EXPA.

CA-ASM2 allows you to change the regular archive unload process to cause selected cataloged objects being archived to be recataloged to a fictitious volser. A fictitious volser does not identify any actual volume. Data sets cataloged to a fictitious volume are assumed to be archived or unavailable on disk.

Note: MIGRAT should not be specified when recataloging a data set to a fictitious volser. Some IBM software expects data sets recataloged to volsers with MIGRAT to have been archived with DFHSM, and takes appropriate action.

This facility provides required support for archive processing of disk data sets that are part of a Generation Data Group. Such data sets should not be uncataloged at archive unload time because this effectively deletes them from the GDG and disrupts the user's GDS enumeration process. Therefore, when CA-ASM2 archives a GDS, it can be recataloged to a fictitious volser to preserve its absolute position in the GDG. When

IXR reloads a GDS, the data set is again recataloged to the disk on which it was reloaded. The use of a fictitious volser is mandatory for proper IXR support of GDG disk data sets.

Using the fictitious volser facility also serves as a quick indicator that CA-ASM2 archived a given (cataloged) disk data set. For example, this allows LISTCAT output to indicate a data set's archive status and catalog information. Since the fictitious volser is defined to archive runs and IXR supports the use of multiple fictitious volsers, your installation can use varying volsers associated with different archive passes as an administrative mechanism. For example, you might have one fictitious volume for voluntarily archived data sets and another for those archived involuntarily during archive system pass.

Take care in deciding whether to use the fictitious volser feature for non-GDG data sets. In particular, if your installation has a large amount of archive activity (involving a large number of cataloged data sets), you must consider the impact of retaining an ever-increasing and potentially huge number of MVS catalog entries. CA-ASM2 automatically deletes fictitious volser catalog entries when archive copies expire. Recataloging to a fictitious volser is beneficial when alias names are used for data sets. Catalog search/reporting functions find the base dsname and aliases showing the fictitious volser. Reloads from archive relate the alias name to the data set on the new volume.

You can also use the fictitious volser facility for both non-VSAM and VSAM data. Archived VSAM data is recataloged as a non-VSAM data set when you use this facility. During reload processing, the non-VSAM type of catalog entry is uncataloged and the object again becomes cataloged as VSAM.

The fictitious volsers must conform to the normal MVS notation (one to six alphanumeric/national characters) and the volsers used must never appear in the system as actual DASD or tape volumes. Computer Associates suggests using names such as ARCHIV, ARCVOL, and so forth. When IXR is up a job never attempts to go through MVS allocation with a fictitious volser (unless the user specified one in JCL). This is because IXR builds a permanent global table of all fictitious volsers during its initialization.

2.11.4 DSA to Tape Copy Duplexing

$COPYTP was designed to execute without needing SYSIN control parameters. For duplexing of archive and backup tapes copied from DSA unloads, $COPYTP continues to not need SYSIN control parameters. The automatic ASM2 mode processing is performed based on the following:

- $COPYTP detects ASM2 mode tape copying by the presence of an LOXXX DD statement.
- $COPYTP detects whether an archive, backup, or full-volume backup run is being made by the next to the last qualifier of the data set allocated to the LOXXX DD statement (ARCH, BKUP, or DEFRAG respectively).

In Version 4.1, $COPYTP was changed to automatically detect whether the run is a disk-to-tape unload or a DSA-to-tape copy run from the format of the last qualifier of the data set allocated to the LOXXX DD statement. Prior to Version 4.1 the format of the last LOXXX qualifier was: `LOxxxxrr`

where:

`xxxx` is USER for explicit archive or backup runs, PERM for permanent archive runs, and SYS for system-initiated runs.

`rr` is a 1- or 2 character run ID, which allows multiple concurrent runs of each type to be made.

In Version 4.1, the format of the low LOXXX qualifier remained the same for disk-to-tape unload procedures but for DSA-to-tape unload procedures the format of the lowest LOXXX dsname qualifier was defined as: `LODxxxrr`

where:

`D` defines the run as a DSA-to-tape rather than disk-to-tape procedure.

`xxx` is USR for explicit archive or backup runs, PRM for permanent archive runs, and SYS for system-initiated archive or backup runs.

`rr` is a 1- or 2-character run ID, which allows multiple concurrent DSA-to-tape copy runs of each type to be made.

2.12 IPC and Journal Space Estimation

If you are upgrading to Version 4.2 from a version earlier than Version 4.0, to determine the intial number of records in the IPC, add the number of records in the backup and archive catalogs together to determine the total number of records that the conversion program converts into IPC records. Use this record count and the chart on 2-36 to compute space requirements.

If you are a new CA-ASM2 user, use the following guidelines to determine the initial number of records in the IPC.

Backup records To estimate the number of backup records in the IPC, multiply the number of DSNs on all DASD volumes to be backed up by $BMINVER + 1. $BMINVER is the minimum number of versions to be kept in the IPC for backup records.

Archive records This value varies according to the value of $CYCLETM in $OPTIONS and your installation standards. Experiment with the number of records that would be archived by running the ASM2SYSA procedure in SIMULATE mode. Adjust $CYCLETM accordingly depending on how aggressive your archiving standards are.

$DEFRAG records If you use $DEFRAG to do full-volume backups, there is one Defrag Volume Record (DVR) written for each DASD volume dumped. In addition, if $DEFRAG is run in ASM2 mode, there is an UNLOAD record written for each data set on the volume being dumped.

Consider the following facts:

Average IPC LRECL=	585
IPC CI Size (CISZ)=	4096
Max. records per CI=	7
Recommended Free Space	
Control interval (CI)=	10%
Control area (CA)=	10%

Free space (FSPC) should be defined to allow for empty space in the control interval and the control area when the cluster is loaded. Since the IPC is a KSDS (Key Sequenced Data Set), this free space is necessary for data records that are updated or inserted during CA-ASM2 production jobs. By specifying FSPC(10,10), 1 record per control interval and 15 control intervals per control area is reserved to be used for record updates and insertions.

The IMBED keyword on the cluster definition specifies to write a sequence-set record for each control area with the data component. This gains some performance but will take up at least another TRK from the control area.

Use the following table to determine how many Intergrated Product Catalog (IPC) records fit per TRK/CYL on various DASD types. All calculations are based on NOIMBED.

Description	NOTE	3380	3390	9345
Records per CI	#1	6	6	6
CIs per TRK	#2	10	12	10
TRKS per CA	#3	13.5	13.5	13.5
CIs per CA	#4	135	162	135
Records per TRK	#5	60	72	60
Records per CYL	#6	810	972	810

Note 1.

The IPC CI size is 4096 and the average LRECL is 585 bytes. In addition, 10 percent CI free space is defined giving 6 records per CI. The average LRECL for IPC records was determined by analysis on several IPCs ranging from fairly small to very large.

```
( 4096 / 585 ) * .90 = 6 records per CI
```

Note 2.

Figures are with a CI size of 4096.

Note 3.

10 percent Computer Associates free space is defined, reserving 10 percent of each Computer Associates as free space.

```
( 15 * .90 ) = 13.5 TRKS per CA
```

Note 4.

With Computer Associates free space defined as 10 percent, 13.5 tracks per Computer Associates are used. There are 10 CIs per track on a 3390.

```
CI/CA = (CIs per TRK * TRKS per CA)
CI/CA = (10 * 13.5 ) = 135
CI/CA = (12 * 13.5 ) = 162
CI/CA = (10 * 13.5 ) = 135
```

Note 5.

To determine records per TRK based on FSPC(10 10):

(Records per CI * CI's per TRK) = records per TRK

```
(6 * 10) = 60
(6 * 12) = 72
(6 * 10) = 60
```

Note 6.

To determine records per CYL based on FSPC(10 10):

(Records per TRK * TRKS/CA) = records per CYL

```
(60 * 13.5) = 810
(72 * 13.5) = 972
(60 * 13.5) = 810
```

2.12.1 Computing Initial Space Requirements for IPC

For purposes of illustration, assume a total of 120,000 records to be initially placed in an IPC. Using the figures from Notes 5 and 6 above, if the IPC is going to reside on a 3390 the formula for cylinder allocation would be:

```
(120000 / 810) = 149 cylinders needed
```

For a 3390:

```
(120000 / 972) = 124 cylinders needed
```

For a 9345:

```
(120000 / 810) = 149 cylinders needed
```

2.12.2 Journal Space Requirements

The CA-ASM2 journal is used to record all update activity since the last IPC backup utility (M2PCBRU) was run. It also contains the data set name and volser of the most recent IPC backup tape. The journal is cleared each time the IPC backup utility is executed by resetting the high used RBA to 0. Since this file should be cleared on a regular basis, it is recommended to run an IPC backup at least once a week. The allocation generally should be approximately 10 to 15 per cent of the total allocation of the IPC.

2.13 IXR Overview

The installation of IXR is accomplished after installing the base product. If you plan to use the threshold driven archival facility, Computer Associates recommends the use of IXR. This may be done immediately after installing the base product or at any later time that it is convenient. An overview of IXR is included in *CA-ASM2 Getting Started* to help you decide if and when you want to install this feature.

Detailed information on the operation and use of IXR is included in the *CA-ASM2 System Reference Guide*. Messages associated with IXR are located in *CA-ASM2 Messages*.

2.14 IXR Environmental Considerations and Restrictions

This topic discusses the aspects of an MVS system that affect IXR operation: catalog environment, data set name distinction, VSAM/non-VSAM distinction, DASD volume accessibility, reload volume conflicts, nonstandard data management actions, IXR staging and SMF exits, and data references that IXR cannot intercept.

2.14.1 Catalog Environment

IXR operates in the default catalog environment of MVS. Cataloged data sets (both VSAM and non-VSAM) subject to IXR processing should be cataloged in the correct CVOL, VSAM, or ICF catalog. Data sets that are accessed only by using JOBCAT/STEPCAT DD statements or the IDCAMS CATALOG command parameter have problems with IXR.

Problems may occur because IXR has no built-in facility to direct its own catalog requests (both locating and cataloging) to a particular MVS catalog. Therefore, the LOCATE issued by IXR during the availability analysis phase for a data set may return a not found indication when, in fact, the data set is simply cataloged in an inappropriate catalog. This causes IXR to mistakenly assume the data set is unavailable and so proceed in error with the reloadability analysis phase. This, in turn, may cause an erroneous unavailable indication or, worse, an attempt to reload an archive when the data set is already on disk.

Do **not** rectify the situation by adding a JOBCAT or STEPCAT to the IXR subsystem procedure JCL. Although this action alleviates the above difficulty with LOCATE by forcing a search of a specific catalog, all CATALOG operations performed by IXR are also forced to the specified catalog causing many problems.

The general rule is that cataloged data sets subject to IXR processing must adhere to the established default catalog hierarchy as indicated by the alias-UCAT-CVOL entries in the system's master catalog. This includes data sets that do not have a high-level qualifier entry in the master catalog but are cataloged in the master catalog.

If the environment in which IXR is executing requires more than the first high-level qualifier of the data set name to correctly catalog the data set into the appropriate user catalog, the RLDTMPNM parameter should be specified in the IXR startup parameters. This parameter causes IXR to use either the first three high-level qualifiers of the original data set name for non-VSAM data sets, or the first two high-level qualifiers of the original data set name for VSAM data sets, when allocating the temporary data set that is used when IXR reloads a data set.

This restriction does not affect noncataloged data sets (non-VSAM data sets referenced with explicit DASD unit name and volser).

2.14.2 Data Set Name Distinction

Multiple distinct data sets with the same data set name are possible in MVS systems. IXR allows name duplication among the IPC records and recognizes each distinct data set or cluster if there are other distinguishing factors. For cataloged data sets, the combination of data set name and four-character SMF system ID (of the archiving system) must be unique. Each system ID indicates a particular master-user-CVOL catalog structure.

The archive unload process can record the SMF system ID with each archive. Archives created before Version 3 archive unload programs were installed do not have an associated system ID. IXR's IPC record for these archives simply indicates no system ID. Only one such archive is allowed per distinct cataloged data set name.

IXR permits and distinguishes the presence of a VSAM cluster and a cataloged non-VSAM data set with the same name and system ID (or no system ID). However, do not rely on this for reasons cited in the topic titled "VSAM/non-VSAM Distinction" following this section.

For noncataloged non-VSAM data sets, the combination of data set name and original DASD volser must be unique. It is impossible to create two data sets of the same name on the same DASD volume. However, in a CA-ASM2 environment, it is possible to create a data set on a volume using the same name as a data set that was archived from that volume. A related discussion is in the topic titled Nonstandard Data Management Actions on 2-42.

IXR selects the record associated with the most recent archive action if the IPC contains multiple archives for the same data set name.

Typically, these records represent different archives of the same data set rather than distinct data sets with the same name. IXR always reloads the most recent archive of a data set or cluster.

2.14.3 VSAM/non-VSAM Distinction

A VSAM cluster and a non-VSAM data set with the same name are possible in an MVS system. They can even be located on the same DASD volume. However, this is undesirable because confusion may result when a user attempts to reference one object or the other.

If a VSAM cluster and non-VSAM data set of the same name are both archived, (perhaps from the same volume), it may be impossible for IXR to choose which archive to reload unless the user's reference clearly indicates the type, VSAM or non-VSAM. For example, in a job-level or step-level staging situation, the presence of the JCL AMP or DCB parameter indicates to IXR whether a VSAM or non-VSAM object is the user's intended access. However, if no such distinguishing factors are present in the JCL, the results depend on the nature of the user's reference and the point of IXR interception.

If the user is referencing through the system catalog (UNIT and VOL are not specified), IXR attempts to use the SMF system ID of the running system as the deciding criterion. If both archives have the same or no system ID, or if neither matches the ID of the running system, IXR cannot choose. In this situation, the user's job ultimately receives a JCL error indicating data set not found.

If the user's reference is not through the catalog (UNIT and VOL are specified), IXR uses the indicated volser as the selection criteria. If both the VSAM and non-VSAM data are archived from the same volume, IXR cannot select. This does not result in a JCL error. Instead, IXR defers any action for the named data set until OPEN time when it is clear whether the access is intended to be VSAM or non-VSAM.

For purposes of IXR implementation compatibility with future MVS enhancements, users should avoid creating multiple distinct data sets and VSAM clusters (or even VSAM cluster components) with the same name.

2.14.4 DASD Volume Accessibility

IXR does not take actions that require access to a DASD volume that is not online. This has an effect on several aspects of IXR operation.

In job-level and step-level staging, the IXR intercepts for these functions actually execute before the user's job goes through MVS device allocation. If the user coded DD statements with explicit UNIT and VOL information, IXR's availability analysis phase determines if the data set is currently on the indicated disk volume. However, if the volume is not logically online to the system, IXR does not attempt to cause it to be mounted. Instead, IXR processing ends for the associated data set as though the data set is available.

The data set, if archived, is not reloaded during job-level or step-level staging. Instead, the user's job goes ahead with initiation and through MVS allocation recovery to mount any required offline volumes. Then, in a later IXR intercept (VSAM or non-VSAM OPEN), IXR becomes active again and reloads the data set or cluster to the now online device.

A similar situation occurs when IXR processes cataloged data sets or clusters during job-level or step-level staging and in the allocation LOCATE intercept. There are some archives, primarily noncataloged data sets, that are required to be reloaded to their original (home) DASD volume. In processing such archives, if IXR finds that a required volume is not online, it bypasses the data move phase of the request and ends the request as though the data set is available. In the case of LOCATE interception, IXR provides MVS allocation with the correct unit and volume information so the user's job can undergo appropriate allocation recovery. Then IXR can reload the data set or cluster in one of the OPEN-based intercepts.

The purpose behind these offline/online restrictions is to avoid having the IXR address space going through allocation recovery and mount processing on behalf of some other job. Since allocation recovery issues a WTOR and waits on an operator reply, this

might severely impact IXR performance. IXR should not suffer the delay, processing, and accountability for allocation recovery when a user's job references offline volumes.

2.14.5 Reload Volume Conflicts

IXR supports installation-controlled selection of reload volume for catalog-type references to cataloged non-VSAM data sets. This is possible because IXR's interception of such references normally takes place before MVS device allocation. There are situations in which this selection process can cause error conditions (conflict over reload volume) for users subject to IXR processing.

Conflict can occur when two separate jobs reference a cataloged non-VSAM data set, and one job has actually specified the data set's unit and volume rather than relying on the catalog. If IXR intercepts the other job first (the one making reference through the catalog), IXR invokes the reload volume selection process and selects an available DASD volume. This could be a different volume than the one specified by the first job. IXR ultimately intercepts the first job and finds that the data set is being reloaded or was reloaded to a different volume. Since IXR cannot change the user's specified allocation, a reload volume conflict situation exists. The job with the specified unit and volume ultimately receives a 213-04 abend.

The best way to avoid this problem is to always reference cataloged non-VSAM data sets subject to CA-ASM2 archive and IXR reload through the catalog (without UNIT and VOL parameters).

2.14.6 VSAM Sphere Processing

All components of a VSAM sphere are processed together during unload and reload operations unless the user requests that only the base cluster be unloaded. The VSAM cluster name, alternate index names and path names are all recorded in the IPC and all may be used to drive a reload of the sphere. If IXR intercepts a reference for any component of a VSAM sphere, the entire sphere is reloaded.

2.14.7 Nonstandard Data Management Actions

CA-ASM2 with IXR provides a storage management environment that operates, as much as possible, like normal MVS without CA-ASM2 IXR. However, IXR and CA-ASM2 do not assume control over all native MVS data management functions to translate these functions into logical storage management operations. In particular, they do not intercept or even observe functions such as non-VSAM new data allocation, scratch, catalog, and rename; and VSAM DEFINE, ALTER, and DELETE. Using these functions with data that is subject to CA-ASM2 archival and IXR-originated reload requires several considerations.

Nonstandard data management actions can conflict with IXR operation. With IXR, a data set or cluster exists validly in exactly one place and its (MVS) catalog status changes only when the data is moved to another place or is logically and physically deleted from the system. When data is archived, IXR expects it to be uncataloged, or perhaps recataloged to a fictitious volser.

It is possible for a user to create a new data set with the same name as one that is archived. Logically, in CA-ASM2's view, the user should receive a duplicate data set indication for this action unless the new data set is distinct from the old under the criteria described in the topic titled "Data Set Name Distinction" on 2-40. No such error indication occurs because CA-ASM2 and IXR do not monitor data set creation/allocation activity. In this situation, the user logically disabled the archived data set for IXR reload purposes. Since a data set of that name now exists on DASD, IXR never finds it unavailable and thus never considers reloading the archive. (However, it may still be possible to reload the archive through specific requests with the CA-ASM2 $RA command.)

Another nonstandard action that can affect IXR operation is cataloging of archived data sets. For example, if the user employs the IEHPROGM utility to catalog (or recatalog from fictitious volser) a data set name that is archived, it becomes possible for references to that data set name to get by IXR's staging and LOCATE intercepts. (LOCATE gives a return code of zero so IXR assumes the data set is available.) The user ultimately allocates the volume to which the data set name is cataloged. If OPEN is ever issued, the absence of a VTOC entry for that name causes IXR to get control. Since a specific volume is allocated, IXR searches the IPC for a data set of the same name that is archived from that volume. If IXR finds none (because the user cataloged the data set name to a volume other than the one it was once on), IXR cannot select an archive and the user ultimately receives a system 213-04 abend.

Another concern is user dependence on archives. Before IXR, users were very aware of the archive process. This led many users to take actions they might otherwise not take knowing that archived copies of data existed. The classic example is the user who reloads a data set or cluster (using the CA-ASM2 $RA command), deletes it from DASD with an MVS SCRATCH or DELETE operation, and then again reloads the same archive to get the data back on disk. In this circumstance, the user is actually using CA-ASM2 archives as a backup mechanism. That is, the user feels free to damage or delete a reloaded copy of the data because the user knows the archive can be reloaded again.

Unfortunately, this action is in direct opposition to the IXR philosophy which says a data set exists in only one place at a time. When IXR reloads a data set to disk, the archive copy used as input is conceptually (though not actually) deleted because IXR knows the one permissible copy is now on DASD. If the user deletes (or renames) the data set and then again references it, IXR normally refuses to perform what is called a secondary reload of the data.

The following discussion and example explains why IXR responds negatively to secondary reload requests. CA-ASM2 with IXR provides an environment equivalent to plain MVS. Regardless of storage management actions, if you delete your data set or cluster, it is gone unless you took steps to ensure that backup copies (not archive copies) exist. This is a data integrity issue. Consider an application where a file of transactions is read in and processed. Suppose that the current transaction file is archived and then reloaded by IXR when the application runs. On completion, the application deletes the transaction file (expecting to have another built for the next processing cycle).

What happens if the application is accidently rerun? In plain MVS (no CA-ASM2 or IXR) the job receives a data-set-not-found JCL error or S213-04 abend and, as far as transactions are concerned, no harm is done. If, however, IXR was present and allowed the secondary reload, the transactions would erroneously process a second time. From the application's perspective this is a logical impossibility because the transaction file was scratched.

A convenience mechanism in CA-ASM2 (where reload activity is always manually requested) becomes a bona fide data integrity exposure with IXR (where reload activity is automatic). This is why IXR normally prohibits secondary reloads.

There are times when it is desirable to force a secondary reload. You can always do this with the $RA command. Also, IXR provides a processing option (SEC, recognized on a request-by-request basis) that indicates permission to do a secondary reload through IXR. Computer Associates recommends that your installation set the default option SEC to indicate no secondary reloads, and consider limiting the ability to reverse the option. To do this, use the installation override capability and the IXR Reload Options Exit, $IXRUROX. (This is the exit's default member name. You can change the name in the IXR startup parameters.)

2.14.8 IXR Staging and SMF Exits

The IXR intercepts for job-level and step-level staging are packaged in the MVS SMF job initiation (IEFUJI) and step initiation (IEFUSI) exit routines. IXR request processing options control the use of these staging features (see the topic titled "IXR Processing Options" in the *CA-ASM2 System Reference Guide*. Your installation controls whether the SMF exits are installed and, if so, whether they are invoked for each of the three address space types (controlled by the MVS SMF parameters in the SMFPRMxx member of SYS1.PARMLIB).

If your installation does not install the exits, there is no IXR staging capability. If one of the two is installed, only that level of staging is available. (Job staging is in IEFUJI and step staging is in IEFUSI.) Staging is a key efficiency feature of IXR, often reducing the number of tape mounts required for reloads and/or performing simultaneous reloads from an archive Disk Staging Area. Only one of the staging intercepts should be activated by default.

If both IEFUJI and IEFUSI are installed, they can be executed only for batch address spaces. Unless you change the system SMF parameters, only batch jobs have access to the IXR staging capability. This is not a problem because started tasks rarely require IXR action.

The job- and step-level staging exits are optional and are not required to ensure archived data sets get reloaded when they are needed. The required IXR exits, LOCATE non-VSAM and VSAM, dynamic allocation, and S213 can be depended on to reload all data sets and clusters when needed.

The purpose of the IXR job- and step-level staging exits is to provide a way of reducing the number of tape and cart mounts needed for reload in some environments.

For example, if the data sets used by a job(s) tend to be archived together and reside on the same archive tapes, the job- ***or*** step-level exit could be used to reduce mounts. Note that the job- and step-level staging should never both be turned on. This would double the overhead as all reloads would occur in job-level staging but step-level staging analysis would still occur.

In environments where very few IXR archive reloads occur, neither the job- nor step-level staging exits need be activated. Experiment with job-level (with step-level off) and see if there is any impact on the number of mounts and system performance. If job-level has a benefit, experiment with job-level versus step-level staging turning one on at a time with the other off. Step-level staging may increase the number of mounts. The choice must be based on your environment.

2.14.9 Data References IXR Cannot Intercept

IXR can successfully intercept and process reload operations for all normal types of data reference. This is true regardless of device allocation (JCL or dynamic), the data type (VSAM or non-VSAM), and the OPEN type (normal or Type J).

However, programs that use nonstandard data management techniques fall outside IXR's reach. In particular, IXR does not process programs that use the IBM LOCATE or OBTAIN macros to determine the accessibility of a data set before using dynamic allocation or OPEN. The LOCATE or OBTAIN for an archived data set normally returns a not found indication. (The LOCATE returns a found indication if your installation is cataloging archived data sets to fictitious volsers.)

2.14.10 Renaming of Non-GDS Data Sets

The initial release of IBM's SMS offering did not allow non-GDS data sets to be renamed to a GDS (Generation Data Set). This prevented IXR reloads of SMS GDS data sets as IXR uses a temporary data set name to download into and then renames the data set only if the reload I/O operation is successful.

IBM for APAR OY51942 has released an SPE (small program enhancement) which provides this renaming capability for renaming data sets as a GDS and then inserting them into the GDG (Generation Data Group). The following PTFs or equivalent are required in DFP environments for IXR to successfully reload SMS GDS data sets:

DFP 3.2 UY90959
DFP 3.3 UY90960

2.15 IXR Operational Considerations

This section contains information on the following as they pertain to IXR:

- CSA requirements and control
- Subsystem program properties
- TSO
- Controlling IXR processing options
- JES
- Volume selection
- Tape device allocation
- IPS and ICS considerations

2.15.1 IXR CSA Requirements and Control

IXR uses control blocks allocated in CSA (extended CSA on MVS/XA) for communicating request information between the subsystem address space and the users address spaces. Although the amount of CSA required is generally small, IXR provides careful control facilities so your installation can manage IXR effectively if CSA availability is a concern.

A base amount of CSA is required just to bring up IXR. This is only about 6000 bytes (plus about 2000 bytes of SQA). The amount of additional CSA needed depends on the amount of concurrent IXR traffic to be processed. First, each distinct address space (job, TSO session, or started task) using IXR at a given moment requires 32 bytes of CSA. Next, for each concurrently active IXR request, 132 bytes of CSA are required to represent the request and 224 bytes are required for each data set being processed. Another way of looking at this is that a single request needs 132+(n*224) bytes of CSA, where n is the number of data sets being processed in the request. The maximum possible number of requests is theoretically limited by the number of tasks (not address spaces) in the system. However, it is reasonable to expect that in your system the maximum number of concurrent IXR requests is much smaller than the number of active address spaces, and it is unusual for a single address space to have more than one active IXR request at a time.

Only requests that originate in IXR's job-level and step-level staging intercepts involve multiple data sets. The expectation is that a typical staging request involves many data sets (as many as there are nontemporary, existing JCL references to disk data sets, which must be looked for in the IPC for reloadability).

IXR does not use individual GETMAIN and FREEMAIN processing to acquire and release the request-related CSA storage blocks. Instead, IXR acquires and globally manages CSA in larger chunks called cell pools.

Note: This cell pool facility is indigenous to IXR. It is not the same as the cell pool architecture used by certain MVS internal functions. A cell pool is basically a

number of pre-GETMAINed control blocks of a given type. Of relevance to IXR request processing (and CSA usage) are three cell pools:

1. Address-space-related blocks (ASBP and ASBS described on 3-53).
2. Request-related-blocks (XRBP and XRBS described on 3-54).
3. Data-set-related-blocks (DSBP and DSBS described on 3-55).

IXR manages cell pools in a manner similar to DASD space. Your installation sizes each cell pool by specifying the following three characteristic numbers:

1. Primary number of cells
2. Secondary cell allocation quantity
3. Number of secondary quantities allowed

When IXR is initialized, each cell pool receives its primary cell allocation (by normal CSA GETMAIN in MVS subpool 241). As intercepts activate in users' address spaces, cells become allocated to requests through a logical IXR process rather than by individual GETMAINs. As requests complete cells are returned to their respective pools. If an intercept requests a cell from a pool that has no available cells, CSA GETMAIN automatically performs a secondary cell allocation provided the maximum allowed number of secondary allocations is not reached. When a cell pool is 100 percent allocated and there are no free cells, IXR requests begin to fail with appropriate reason code information. For CSA cell pools, IXR issues a warning console message (with WTO descriptor code 11) when a cell pool reaches 80 percent of its maximum allocation and another message when the cell pool is exhausted.

Thus, your installation can precisely control IXR use of CSA for each of the three request-related storage blocks. Variation of the cell pool size characteristics tends to produce different CSA use behavior. For example, choosing relatively small primary and secondary allocation quantities but allowing a large number of secondaries causes IXR to acquire CSA gradually and with a greater sensitivity to IXR traffic. Using larger cell allocation quantities with relatively few (or perhaps zero) secondaries allowed minimizes IXR's acquisition of CSA after initial start-up.

Besides the explicit level of control provided for each CSA cell pool, IXR can be instructed to observe an overall global limit on CSA allocation for any purpose. This provides an easy mechanism for placing an absolute ceiling on IXR CSA in your system if CSA is a constrained resource. The global limit takes precedence over all other CSA controls. For example, an attempt to extend a CSA cell pool fails if it causes the global limit to be exceeded, even if the specific cell pool has not exhausted its allowance of secondary allocations.

CSA usage is not the only area where IXR can have an impact on the MVS common-private storage boundary. A certain amount of IXR programming (that which executes in address spaces other than the IXR subsystem) must be installed in the MVS Link Pack Area (LPA). Additions to LPA or MLPA may cause an increase in commonly addressable storage and a commensurate decrease in available private area storage.

2.15.2 IXR Subsystem Program Properties

IXR is required to execute as an APF-authorized process because of the direct use of restricted facilities such as allocation-style superlocate and CSA subpool GETMAINs. Programs invoked by IXR intercept code are preloaded by the subsystem during initialization yet must be accessible to all MVS address spaces. All such programs are both reentrant and refreshable, and therefore conform to established LPA conventions on all MVS systems.

Your installation may want to consider other optional IXR properties that can improve IXR's performance, reliability, availability, and serviceability. An obvious issue is swappable versus nonswappable execution. Computer Associates recommends you make the IXR address space nonswappable. This is because IXR has certain activities that execute at timed intervals on the scale of minutes. If IXR is swappable, it tends to undergo constant regular swap-out and swap-in during otherwise idle periods, which may be undesirable. You can establish nonswappable execution either by the use of a particular MVS SRM domain or by putting the name of IXR's main subsystem module, $IXRMAIN, in the MVS Program Properties Table and assigning the nonswappable attribute to $IXRMAIN.

Note: Detailed information on the Program Properties Table is provided in the appropriate IBM manual.

Another optional IXR property is the assignment of a unique storage protect key to the subsystem. Unless otherwise directed, IXR executes in the MVS default user protect key of 8. All IXR CSA is GETMAINed with this key and all IXR intercepts switch to this key while executing. A disadvantage to this is that any normal user program, which also executes in key 8, can erroneously overlay IXR data in CSA and thus cause errors in IXR execution or even IXR termination.

Using the Program Properties Table, you can assign IXR a storage protect key other than 8. This makes it impossible, or at least a lot more difficult, for another program to accidently damage IXR information. While your initial reaction might be to assign key 0 to IXR, Computer Associates does not recommend this. Doing this causes IXR and all IXR intercepts to execute in key 0, making it much easier for an IXR problem to cause overlays of other system storage and even possibly crash the system. The best approach is to assign one of the available (reserved) protect keys to IXR. Likely candidates include keys 3 and 4, key 2 (if the VSPC product is not in use), and key 7 (if IMS is not in use). You might also consider keys 9-15, but remember that these keys are also used by V=R address spaces. The strategy is to assign IXR a key that is never used by anything else. This makes it almost impossible for a program or the system to step on IXR, or conversely for IXR to step on them.

Finally, you can use the Program Properties Table to skip the MVS password request to the operator for every data set that IXR wants to reload. Skipping the password check saves the operator time and effort. IXR maintains security by transferring password protection information from the IPC record to the DSCB. To skip the password check, set the X'02' attribute for $IXRMAIN if performing a PPT zap or

NOPASS if applying PPT updates through the SCHEDxx member of SYS1.PARMLIB (XA 2.2 systems and later only).

2.15.3 IXR and TSO

IXR operates correctly with all three types of MVS address space: batch, started task, and TSO. Your installation may want to limit or even disallow IXR activity for TSO users for the reasons discussed in the following paragraphs.

First, IXR is a realtime reload facility. When IXR intercepts a reference to an archived data set, the associated process waits until IXR processing completes. If the archived data is in the DSA, the reload is much faster than it would be from tape. With archives on magnetic tape, if the process is a TSO user's command or CLIST, a considerable delay may result waiting for the tape to be mounted and data moved. It is unrealistic to expect a TSO user to wait from 15 minutes to an hour or more for completion of a reload. Your installation may consider it undesirable to have a terminal logged on for such a period doing essentially no productive work.

You can deal with this issue in several ways. One way is to speed reload processing to a level tolerable to interactive users. The only realistic way to do this is to eliminate human delay from the tape access process. This means using an automatic tape library or perhaps a distinct product such as the Masstor M860. An automatic tape library (ATL) is a device that automatically selects and mounts conventional tape reels. The Masstor M860 simulates tape operation using mass-storage-style cartridges.

Another way is to allow TSO users to schedule IXR reloads for asynchronous execution. The ability to request asynchronous IXR reload processing is an option (see Option: Interactive Asynchronous Requests in the chapter "Intelligent Transparent Restore (IXR) of the *CA-ASM2 System Reference Guide*) associated primarily with interactive mode IXR users. If your installation allows it, this facility is built into the interactive dialog process. When IXR intercepts an interactive user and finds that a reload is required, it informs the user and permits the user to reply ASYNC and the usual GO, CANCEL, or IGNORE response. If the user enters ASYNC, IXR schedules the associated reload and then allows the intercepted process to proceed (and, usually, fail due to the data set still being absent from disk). IXR processes scheduled IXR reloads like any other except that, for TSO users, a SEND command informs the user when the reload is completed.

The ASYNC reload facility allows IXR to operate somewhat like the CA-ASM2 queued reload facility. The differences are that the operation is triggered by normal data management action rather than by an explicit command, and the reload is actually processed through IXR and thus may be considerably faster. The issue you must consider is whether IXR is to be used by TSO-originated asynchronous traffic at all, or if the queued reload facility is required for TSO users. Your installation may want to focus primarily on batch/STC IXR processing, at least initially, and perhaps consider introducing TSO work later.

If your installation lacks an ATL or M860 and does not want TSO asynchronous reloads processed in IXR, the final choice is to disable IXR for time-sharing address

spaces. You can do this by using processing option defaults and overrides. For more information, see IXR Processing Options in the chapter "Intelligent Transparent Restore (IXR)" of the *CA-ASM2 System Reference Guide*.

2.15.4 Controlling IXR Processing Options

IXR has a variety of options that are determined and observed on a request-by-request basis. This topic discusses how these options are controlled. Individual options and their meanings are discussed in IXR Processing Options in the chapter "Intelligent Transparent Restore (IXR)" of the *CA-ASM2 System Reference Guide*.

Your installation is responsible for the first level of control. It specifies the IXR system startup parameters which specify the initial default options that apply to IXR requests. Separate default specifications are made for each of three MVS space types: batch, TSO, and started task. An IXR request's options are simply the defaults for the type of address space in which IXR invokes the originating intercept. The IXR startup parameters are distributed in member IXRPARM of the CAI.L142.PARMLIB data set.

The next level of control lies with the user. Through the use of reserved-name DD statements (normally coded as DUMMY), the user can modify the default options that are set for the user's IXR activity. The option DDs can specify job, step, or immediate scope. An option specified as job level must be coded in the JCL for the first step and applies to all steps in a job unless overridden. A step-level option applies to a single job step and overrides any corresponding job-level option specification. Finally, an immediate level option overrides both job and step-levels after the step starts executing. Internally, immediate level options are processed just like step level options. Their value lies primarily in the TSO environment, where they can be dynamically allocated and freed to affect the way a single command or series of commands interacts with IXR. They can be used in batch or started tasks also.

After IXR scans all user option DD specifications and uses them to modify the system defaults, it applies an installation override facility. As with the default settings, it provides separate overrides for batch, TSO, and started tasks. The installation override capability allows your installation to forcibly set or reset IXR options and thus control their use by the end user. In terms of implementation, an override has two processes. First, the option list is logically ANDed with a set of options, then the list is ORed with another set.

To force an option off, the corresponding entry in the AND set should be off. To force it on, the entry in the OR set should be on. The AND string itself defaults to all on and the OR string to all off, effectively allowing the user to modify all system default options.

When option control must be based on a finer distinction than just the batch/TSO/STC distinction, an exit interface is available. IXR invokes $IXRUROX, the IXR Reload Options Exit which runs in the intercepted user's address space, after all the foregoing option-merging is done. This exit examines and modifies the options for each request.

2.15.5 IXR and JES

IXR supports both JES2 and JES3. Under JES2, IXR initially selects a disk volume for reloading an archived data set in the same way as CA-ASM2. It selects according to the following fields in $OPTIONS:

$HOMEVOL

$RLDGRP

$SELDISK

$SOFTWAR (bit X'80')

$TYPEREQ

$VOLGRP

Once chosen, the target reload volume for a requested data set is valid for a length of time set by the $CVLEXP field of $OPTIONS (JES3 only, this field does not apply to base CA-ASM2).

IXR can be set up so that it next invokes the Volume Selection Exit, $IXRUSVX. This exit can alter the choice of reload volume according to any criteria that you want. The Volume Selection Exit is described briefly in the subtopic IXR Volume Set in IXR Operational Considerations on 2-53 and in detail in the *CA-ASM2 System Reference Guide*.

Under JES3, IXR selects a target reload volume in the following way. When JES3 reads the JCL for a job that requests a data set, the JES3 Converter-Interpreter issues an SVC 26 LOCATE for the data set. If the data set is online, the MVS module IGC0002F returns the volser(s) and device type(s) of the DASD volume(s) containing the data set. If the data set is not online, IGC0002F returns a data set not found.

IXR supplies a JES3 LOCATE Intercept to handle the case when a data set is not online because ASM2 archived it. This intercept is described in "JES3 LOCATE Intercept" in the *CA-ASM2 System Reference Guide*. The intercept is packaged and installed with others in a single piece of code called the IXR Catalog Management Intercept, $IXR26DU.

$IXR26DU replaces the CSECT IGG026DU in module IGC0002F. (IGG026DU receives SVC 26 LOCATEs, passes them to another CSECT within IGC0002F, and returns the results to JES.) If $IXR26DU is installed, the JES3 LOCATE Intercept is enabled, and IGC0002F returns a data set not found. Then, IXR searches the IPC for the name of the requested data set. If the data set is not cataloged there, IXR returns a data set not found.

If the data set is in the IPC, IXR returns a list of DASD volumes to JES3 CI. Later, IXR reloads the requested data set to one of these volumes. The following steps detail this process:

1. $IXR26DU intercepts an SVC 26 LOCATE from JES3 CI.

2. If the data set is not online but is listed in the IXR catalog, $IXR26DU passes to JES3 CI the DASD volsers of up to five candidate reload volumes with their device types.

   - If a reload request for this particular data set occurred before, the data set's IPC record contains the list of candidate reload volumes selected for that request.

     When the earlier candidate volume list was selected, the date and time were recorded in the catalog record. The earlier list expired if a set number of days elapsed since that date and time. The $CVLEXP field of $OPTIONS sets the number of days (see 3.5, “Options ($OPTIONS) Affecting IXR” on page 3-47).

     If the candidate volume list is not expired, IXR returns it to $IXR26DU which passes it to JES3 CI.

     If a request for this data set did not occur before or if it did but the old candidate volume list expired, IXR selects a new one.

   - IXR selects five volumes from the master Candidate Volume List, or four plus the home volume. It chooses the four or five volumes from the master Candidate Volume List in round-robin fashion: the first four or five for the first list, the next four or five when a new list is needed, and so on. (You specify the master Candidate Volume List using the CVOL parameter.) See the topic titled "IXR Startup Parameters" in either Chapter 3 or 4. Whether the candidate volume list for an individual data set includes the home volume depends on the setting of $CVLFLG, another field in $OPTIONS.

   - Before returning the five candidate reload volumes to $IXR26DU and JES3 CI, IXR can invoke either the Candidate Volume Selection Exit, $IXRUCVX, or the Volume Selection Exit, $IXRUSVX, or both ($IXRUCVX first). These exits can change the list of candidate reload volumes selected by IXR. For more information, see the topic IXR Installation Exits in the *CA-ASM2 System Reference Guide*.

   Note: IXR invokes these exits only once per data set during each period of validity for each successive candidate volume list in the data set's catalog record.

3. JES3 CI passes the final list of candidate volumes to JES3 SETUP.

4. If the list is not changed by one of the exits described above when the job is initiated, IXR reloads the requested data set to either the home volume or the volume with the largest single free extent depending on the setting of $CVLFLG.

2.15.5.1 Required JES3 Modification

JES3 must have the following job control statements added to its procedure and JES3 recycled:

```
//X$JIXRY DD DUMMY
//X$J26JY DD DUMMY
//X$JDFRY DD DUMMY
//X$JQUIY DD DUMMY
```

The X$JIXRY statement activates IXR for JES3 even though IXR interception of started tasks might be deactivated by default.

The X$J26JY statement activates the JES3 LOCATE intercept which only occurs in the JES3 address space.

The X$JDFRY statement defers any reloads until job execution time while allowing IXR to pass back to JES3 the necessary candidate volumes selected for reload so the job can be scheduled by JES3.

The X$JQUIY statement, although recommended to not be used, should be used for JES3. This prevents any heavy IXR message load of IXR request processing messages due to JES3 requests.

2.15.6 IXR Volume Selection

IXR allows selection of reload DASD volume for catalog-type references to cataloged non-VSAM data sets and VSAM clusters. Your installation controls the volume selection process by essentially the same mechanisms as in regular CA-ASM2 reload processing. It uses the $OPTIONS parameter module for JES3 plus, if necessary, a Volume Selection Exit, $IXRUSVX, and a Candidate Volume Selection Exit, $IXRUCVX. (These are default member names. You can change the exit names in the IXR startup parameters.) Your installation may, if appropriate, set up a distinct $OPTIONx module for IXR and include an appropriate $OPTIONx DD statement in the IXR execution JCL. IXR supports all $OPTIONS volume selection techniques. All these alternatives are described in more detail in 3.5, “Options ($OPTIONS) Affecting IXR” on page 3-47.

Reload volume selection can be tricky in a shared DASD environment if not all DASD volumes are fully shared among CPUs. Be careful to arrange the Volume Selection Exit so that IXR does not reload a data set to a volume that is inaccessible to another processor that might also reference that data set.

2.15.7 IXR Tape Device Allocation

IXR's ability to provide reasonable reload request turnaround depends in part on the availability of tape devices on which IXR can access archive tapes. (It also depends on the frequency of reload requests and on the extent to which multiple reload requests occur for data sets on the same archive tape.) The IXR startup parameters and operator commands provide controls that allow effective tape device management.

IXR is assigned two groups of tape devices: permanent and temporary. Permanent tape devices are allocated to the IXR address space at startup time and are kept allocated the entire time IXR is up regardless of whether they are actually in use for reloads. Temporary tape devices are acquired as needed (when there are reload requests for archive tapes other than those now in use) and are unallocated after use. By sizing these two groups of tapes your installation can exert reasonably precise control over IXR tape device usage. It can effect spontaneous control from hour to hour through operator commands that modify these tape device groups.

Unless you expect IXR reload traffic to be small and widely interspersed, allot at least one tape device to the permanent group. Even though it may sit idle at times, it is always available when a reload request arrives. Temporary tapes are not a guaranteed resource. Regardless of the size set for IXR's temporary tape group, IXR can acquire a temporary tape only if one is available (online and not allocated to another address space).

Note: IXR supports permanent tapes only in MVS systems at the MVS/SP 1.3.2 version and level or higher.

2.15.8 IXR IPS and ICS Considerations

IXR's ability to provide responsive service when MVS is heavily loaded can be improved with suitable IPS and ICS definitions. IXR should be assigned a high priority to ensure it can respond as quickly as possible to intercept reload inquiry requests. However, sometimes a system function attempts to allocate a data set which does not exist and cannot be reloaded (for instance an SPF edit for a recovery data set) and IXR needs to determine this and respond to the calling address space as quickly as possible. A performance group such as one with DP=F9 is a good candidate in which to run IXR.

2.16 IXR Processing Steps

This section reviews the steps for the following processes:

- IXR Intercepts
- IXR Subsystem
- CA-JCLCheck IXR Request Flow

2.16.1 Steps by IXR Intercepts

IXR follows these processing steps when a TSO command or batch job references a data set:

1. a) Optional intercepts (UJI, USI): CA-ASM2 performs availability analysis (described on 2-56) in the initiator address space for all data sets referenced with DISP=SHR/OLD/MOD by the job's JCL or step's JCL. Only data sets not found in the MVS catalog or cataloged to a fictitious volser, or not found on a specified volume, are passed to IXR for possible reload processing.

   b) Required catalog locate intercepts (26A, 26V): These intercepts perform availability analysis on one data set at a time. They only call IXR if the requested data set is not MVS cataloged or is cataloged to a CA-ASM2 fictitious volser.

   c) Required dynamic allocation intercept (S99): This intercept performs availability analysis on one data set at a time. It only calls IXR if a data set is not available on the selected volume and is not miscataloged to the selected volume.

   d) Required OPEN abend exit (213): This intercept receives control during data set OPEN processing only if a data set cannot be found on a volume, and it always calls IXR.

   If IXR is to be called, the intercept continues processing with Step 2 while the intercepted job or command waits. Otherwise, the intercepted job or command continues normally. When an intercept occurs for one task of a multitasking application, only the intercepted task waits for IXR reload processing.

2. The intercept establishes the IXR processing options for the IXR request. It sets default options (determined by your installation) based on whether it is processing a batch job, started task, or TSO command. The user can override some of these options by using option DDs. Some options can disable IXR request processing. After the options are finalized, IXR determines if processing is to continue. If not, the intercepted job or command resumes normal execution with no IXR action, otherwise the intercept continues processing with Step 3.

3. When the job, TSO, or started task (STC) is not in interactive mode, the intercept continues processing with Step 4. If the job, TSO, or started task (STC) is in the interactive mode, the intercept submits an inquire request to the IXR subsystem. The inquire request determines the status of each data set whose name was supplied by the intercept. The status of each data set is one of the following:

   - **Available:** Data set is accessible on DASD and requires no IXR action.
   - **Reloadable:** Data set is not on DASD but IXR identified an archived copy it can reload.

- **Unavailable:** Data set is not on DASD and IXR cannot select a copy for reloading.

If any of the data sets involved in the request have reloadable status, their names display at the TSO terminal and the user can indicate by reply whether IXR should perform the actual reloads. Possible replies are GO, IGNORE, CANCEL, or (if allowed by your installation) ASYNC. It is also possible for a batch job or started task to use IXR's interactive mode. In this situation, the dialog just described takes place at the system operator console rather than at a TSO terminal.

4. The intercept builds and submits a reload request to the IXR subsystem for requests receiving a GO or ASYNC response and for all noninteractive mode requests.

2.16.2 Steps by IXR Subsystem

1. Each data set associated with the reload request goes through the following analysis phases:

   - **Availability Analysis:** Determines whether the data set is available on DASD. If available, IXR flags the data set *available* and performs no further processing for it. The dynamic allocation intercept performs availability analysis and only calls IXR if a data set is not available and not miscataloged to the selected volume. For job-level and step-level staging, a reference of a GDG base name is detected and availability analysis performed for each associated Generation Data Set (GDS).
   - **Reloadability Analysis:** For data sets not available, IXR accesses the Integrated Product Catalog (IPC) to determine if an archive exists. If it selects a valid archive data set, it flags the data set as *reloadable*. Otherwise, it flags the data set *unavailable* and no further processing occurs for it.

     For disaster recovery situations, when the BACKUPS(YES) startup parameter is specified, IXR accesses the IPC to determine if an archive or backup exists. If it selects a valid archive or backup data set, it flags the data set as *reloadable*. If an archive and a backup are selectable, the most recent unload is selected. Otherwise, it flags the data set as *unavailable* and no further processing occurs for it.
   - **Reload Volume Determination:** For data sets for which IXR selects a reload, it establishes the DASD volume to receive the reload. For data sets referenced with explicit unit and volume (not using the MVS catalog), IXR uses the indicated volume. (In this case, the specified volume must be the one from which the data set was archived.) With catalog-type references to cataloged non-VSAM data, IXR invokes an installation-controlled volume selection process. For cataloged data sets referenced through the catalog, the volume selection process skips any volume containing a noncataloged data set with the same name.
   - **Reloading:** IXR allocates the DASD space and reloads the archive from tape. When the operation completes, the data set is on disk and is correctly cataloged just as it was before CA-ASM2 archival. During reloading, the data set has a temporary name generated by IXR. This avoids the problem of other

users or jobs accessing the data during the reload. When the reload finishes successfully, IXR renames and recatalogs the data set as necessary to complete the operation. IXR builds a reload information cell with IXR status fields in the IPC, providing an indication in the archive record that the data set was reloaded by IXR.

2. Finally, when all data sets associated with an IXR request complete these steps, IXR sends the completion status of the data sets back to the IXR intercept. This status information displays at the user's terminal or at the system console for batch jobs or started tasks. Results that display at the console also appear in the printed job log. Finally, the intercepted function proceeds normally or fails depending on the success of the IXR request.

2.16.3 CA-JCLCheck IXR Request Flow

IXR processing follows these steps when CA-JCLCheck calls IXR:

1. IXR establishes the processing option for the request. Using the default options for batch jobs (determined by your installation), the request continues if IXR is active for batch jobs and the IXR intercepts are also on.
2. IXR builds an IXR inquire request and submits it to the IXR subsystem. The IXR inquire request determines the status of the data sets (see IXR Processing Steps in this chapter).
3. IXR returns the status of the data sets to CA-JCLCheck.

2.17 IXR Intercept Drivers

Three IXR intercepts may be installed with driver programs should the user need to invoke additional exits at these processing points. These are the IEFDB401, IGG026DU and IFG0EX0A intercepts, all of which are required and installed as USERMODs.

When CAIRIM installs these intercept points, the CAISMFI Dynamic SMF Interceptor program (a component of CAIRIM) acts as the driver. CAISMFI invokes the SMF exits in the order in which the products are specified in CAIRIM initialization control statements (member CARIMPRM of data set CAI.PPOPTION). Any user-written exits or those for other products are invoked after those defined to CAIRIM. Therefore, no relinking of exit drivers is required since CAIRIM is used to install these intercepts.

If you have specific needs that require explicit ordering of processing in this exit, a driver is provided for IEFDB401. For the dynamic allocation routine IEFDB401, the driver is linked with the IXR intercept and an additional exit if required. The IEFDB401 intercept may only be installed as a USERMOD to the operating system.

2.18 Allocation Manager Overview

The installation of Allocation Manager is accomplished after installing the base product and IXR, if that component is to be used. This may be done immediately after installing the base product and IXR or at any later time that is convenient. CAIRIM must be installed prior to installing Allocation Manager. The installation procedures for CAIRIM are provided in *CA Common Services Getting Started*.

Before installing the Allocation Manager component of CA-ASM2, careful consideration should be given to how Allocation Manager is used. The following should be resolved before building the Allocation Manager Table (AMT).

1. What are the job groups to be defined as a unique type/subtype? Most installations already have groups defined, whether explicit or not. Identify the main processing groups and break these down into subgroups if necessary.
2. How are these groups identified? Reliable conventions for job naming or the use of specific account codes eases the definition process. Allocation Manager may be used to enforce these conventions.
3. What are the target units for the different data set types (temporary, nontemporary...) within each type/subtype group?
4. Is there any data set type for which allocation should not be permitted? By specifying FAIL for the UNIT override value for that type, allocation of that type is not permitted.
5. Are allocations for a particular data set type not to be overridden? By specifying ALLOW for the UNIT override value for that type, no UNIT override takes place.
6. What resources should be restricted for the type/subtype groups? Are there exceptions to the restrictions?
7. Any volumes to be accessed by the DASD group should be mounted with the SYS1.PARMLIB VATLSTnn attribute of STORAGE.

2.18.1 Group Name Modification

The installation of Group Name Modification does not require quite as much planning as Allocation Manager; however, it should be planned carefully. The following should be performed before running Group Name Modification:

1. Identify all DASD groups to be redefined. If additional groups are required an Eligible Devices Table (EDT) generation may be necessary to define them.
2. Identify the volumes that are to be defined for each DASD group. Any groups that are already correctly set up do not require processing by Group Name Modification.
3. Normally, any volumes to be accessed by the DASD group would be mounted with the SYS1.PARMLIB VATLSTnn attribute of STORAGE, however, JCL parameters may be specified to allow PUBLIC and PRIVATE volumes to be controlled as well.

4. Run Group Name Modification in test mode to produce reports of the planned DASD groups for evaluation.
5. Add Group Name Modification to your IPL procedure if desired.

2.18.1.1 TSO Allocation Manager DAIR Component

The TSO Allocation Manager component is independent of the remainder of Allocation Manager. This component, set up during installation, is a single module which includes a table of unit overrides for TSO only allocations. This module is linked together with the IBM Dynamic Allocation Interface Routine (DAIR) as a front end to DAIR. It inspects the DAIR parameter list for requests for the creation of new DASD data sets and overrides the UNIT parameter if no volser is requested. A separate UNIT override value is provided for temporary allocations, nontemporary data sets prefixed by the user's TSO user ID and nontemporary data sets with the TSO user ID as high-level qualifier. It does not reference the Allocation Manager Table (AMT) to establish UNIT override criteria and does not check for access to restricted data sets, units or volumes as does the IEFUJV exit. This component is optional.

Note: ISPF 2.3 and later no longer interface with DAIR for new data set allocations. The TSO Allocation Manager component does not receive control for ISPF allocations at these levels.

2.18.2 Operational Considerations

Both Allocation Manager and Group Name Modification issue WTORs to the system console, thus operations should be notified of the additional messages that are issued at IPL time (or whenever Group Name Modification is run).

1. Allocation Manager issues message CAAM001D to allow the installation the option of coming up with an alternate Allocation Manager Table (AMT) or coming up without any Allocation Manager processing. See message CAAM001D in the *CA-ASM2 Message Guide* for details on the possible responses.
2. Group Name Modification issues WTOR message AMEGNMP-15 before continuing with a production run of Group Name Modification that actually updates the EDT. A response other than U causes the EDT not to be updated (resets to a TEST run). Since it is suggested that modification of the EDT not be attempted while device allocation is in process, this message also gives the operators an opportunity to temporarily halt device allocation while Group Name Modification is running. (Satisfy all outstanding mounts and drain the initiators if not running at IPL.)

2.18.3 Sample Implementations

2.18.3.1 Allocation Manager Sample

A manufacturing company chose to use Allocation Manager to manage allocation and job naming conventions on their system. The following conventions were established before building the Allocation Manager Table and IPLing Allocation Manager:

1. The three main user groups on the system are assigned two-character identifiers for their production jobs; Accounting (AC), Payroll (PR) and Manufacturing (MF).
2. All production jobs are required to begin with P as the first character of the job name, all test jobs must begin with T and all system jobs must begin with S. All jobs not conforming to this convention are to be failed.
3. Each user group has its own designated set of DASD volumes. ACDASD is the group name for the DASD volumes for Accounting, PRDASD is the group name for Payroll, and so forth.
4. Temporary space requests are directed to the set of packs defined by the WORKDA group name.
5. The manufacturing group has a separate set of packs for test and TSO jobs, identified by the MFTESTDA group name. All other test job allocations are to be overridden to the TSODA group name.
6. The manufacturing group is allowed to also use the TSODA group for testing.
7. The system group is to have no limits as to the units to which they may allocate.
8. No test or production job should be able to allocate to a system pack, all of which begin with S or D.

The following list of DASD volsers is assumed to be available in the sample data center's installation:

Volser	Mount Attribute	Volser	Mount Attribute	Volser	Mount Attribute
ACD001	Storage	MFD004	Storage	WORK02	Storage
ACD002	Storage	MFD005	Storage	WORK03	Storage
ACD003	Storage	MFD006	Storage	PUB001	Public
ACD004	Storage	MFD007	Storage	PUB002	Public
PRD001	Storage	MFD008	Storage	SYSRES	Private
PRD002	Storage	TSO001	Storage	SYSRS2	Private
PRD003	Storage	TSO002	Storage	SYS001	Private
MFD001	Storage	TMF001	Storage	DLB001	Private
MFD002	Storage	TMF002	Storage	DLB002	Private
MFD003	Storage	WORK01	Storage	DLB003	Private

The following control statements are used to build the Allocation Manager Table:

```
JOBTYPE=(PROD,1,JOBNM,1,1,PAC)
JOBTYPE=(PROD,2,JOBNM,1,1,PPR)
JOBTYPE=(PROD,3,JOBNM,1,1,PMF)
JOBTYPE=(TEST,3,JOBNM,1,1,TMF)
JOBTYPE=(TEST,4,JOBNM,1,1,T)
JOBTYPE=(SYS,1,JOBNM,1,1,S)
JOBTYPE=(UNKNOWN,1)
UNIT=(PROD,1,WORKDA,ACDASD,ACDASD,ACDASD)
UNIT=(PROD,2,WORKDA,PRDASD,PRDASD,PRDASD)
UNIT=(PROD,3,WORKDA,MFDASD,MFDASD,MFDASD)
UNIT=(TEST,3,WORKDA,MFTESTDA,MFTESTDA,MFTESTDA)
UNIT=(TEST,4,WORKDA,TSODA,TSODA,TSODA)
UNIT=(SYS,1,ALLOW,ALLOW,ALLOW,ALLOW)
UNIT=(UNKNOWN,1,FAIL,FAIL,FAIL,FAIL)
ALLOW=(TEST,3,UNIT,TSODA)
RESTRICT=(PROD,1,volser,S*,D*)
RESTRICT=(PROD,2,volser,S*,D*)
RESTRICT=(PROD,3,volser,S*,D*)
RESTRICT=(TEST,3,volser,S*,D*)
RESTRICT=(TEST,4,volser,S*,D*)
```

While testing the table during installation, the following control statements were also included to cause failing jobs to FLAG errors, but to go on and execute:

```
ERRORS=(PROD,1,FLAG)
ERRORS=(PROD,2,FLAG)
ERRORS=(PROD,3,FLAG)
ERRORS=(TEST,3,FLAG)
ERRORS=(TEST,4,FLAG)
ERRORS=(SYS,1,FLAG)
ERRORS=(UNKNOWN,1,FLAG)
```

2.18.3.2 Group Name Modification Sample

Group Name Modification could be used to define the DASD group names used in the preceding example. If we assume the most recent SYSGEN, IOGEN or EDTGEN specified the group names ACDASD, PRDASD, MFDASD, TSODA, MFTESTDA and WORKDA, Group Name Modification can be invoked to define the contents of each group by specifying the following control statements:

```
ACDASD   ACD001,ACD002,ACD003,ACD004
PRDASD   PRD001,PRD002,PRD003
MFDASD   MFD001,MFD002,MFD003,MFD004,MFD005,MFD006,MFD007,MFD008
TSODA    TSO001,TSO002
MFTESTDA TMF001,TMF002
WORKDA   WORK01,WORK02,WORK03
```

All volsers specified are assumed to be online and mounted with the STORAGE attribute.

2.18.4 Preinstallation Considerations

Each of the Allocation Manager system modifications are installed with the CAIRIM product installation utility. Each step presents the job to modify and run for each installation technique.

- Verify that adequate disk space is available for the Allocation Manager libraries.
- One Type 3 or Type 4 SVC is required before Allocation Manager is installed. Be sure the selected SVC number is available before beginning.
- An Allocation Manager Table (AMT) should be built before the first IPL. (Sample JCL to build an AMT is provided in SAMPLIB member AMBUILD or task AM42IK builds a null AMT.)

2.18.4.1 Reinstallation Considerations

If your installation already has Allocation Manager installed from CA-ASM2 Version 4.0 or later, you do not need to reinstall the system modifications, however you may choose to do so to utilize the CAIRIM installation technique. If you choose not to reinstall the system modifications (IEFUJV, the TSO DAIR modification and the Allocation Manager SVC), you may omit these steps. You must zap your current SVC number into the new versions of the Allocation Manager modules in CAI.CAILIB. Review the other Allocation Manager installation steps to ensure that the existing modules are in place and that the SYS1.PARMLIB updates are retained. There have been no modifications to the format of the Allocation Manager Table; existing tables perform identically with the new version.

Chapter 3. SMP/E Installation Planning

This chapter is to be read and the associated worksheets in Appendix A filled out if you are installing Version 4.2 of CA-ASM2.

Installation parameters and variables are presented in this chapter in the order that you encounter them during the installation. The corresponding worksheets in Appendix A follow the format of this chapter as follows:

- Base Installation parameters
- $OPTIONS field descriptions
- General Protect Criteria
- IXR Installation parameters
- IXR Startup parameters
- Allocation Manager parameters

If you are not going to install IXR and/or Allocation Manager right away, there is no need to complete the associated worksheets at this time.

3.1 Base Installation Parameters

The installation parameters are used to modify JCL to install CA-ASM2. These parameters are edited during the installation of the base product to fit your installation's unique environment. Only one set of parameters is used for all operating systems.

Use the Base Installation worksheet on A-2 to fill in the parameters that you desire for your system(s). A description of each of the parameters follows. You can use the worksheet to specify your selections during the installation.

Two jobs are provided in the installation sample JCL library SAMPJCL to assist the user in determining the maintenance level of operating system modules modified by CA-ASM2. Member L1MODS of SAMPJCL lists the SMP/E information for those system modules modified by the base product installation, IXR and Allocation Manager. The user need only provide the correct MVS SMP/E procedure name and MVS Target zone name. A second job, L1DUMPT, is provided to invoke IBM utility AMASPZAP to perform a DUMPT function for those modules which must be manually inspected to establish the offset into which a table entry must be zapped.

Installation parameters for the base installation are described following. Installation parameters for the IXR portion of the installation are described later in this chapter.

The first 33 parameters are for use with the ISPF edit macro L1CHANGE, distributed in SAMPJCL to globally change variables used during the installation. The remainder of the parameters must be entered manually as they normally only apply to a single task. The parameters are described in the sequence that you specify them during the installation.

3.1.1 JOBCARD Variables

ACCT

Provide the job card accounting fields that you want to replace the default string of JOBACCT. The example format is XXXX,YYYY. Any valid accounting field may be specified.

PROGM

Provide the programmer's name for job card.

JOB2

Provide additional job card fields for job card continuation line. If continuing job card, use comma on end for continuation.

JOB3

Provide additional job card fields for job card continuation line, JES parameters, routing information, and so forth. If continuing job cards, use comma on end for continuation.

JOB4

Provide additional job card fields for job card continuation line, JES parameters, routing information, and so forth.

3.1.2 Computer Associates Common Data Set Prefixes

CAINODE

Provide the prefix for Computer Associates common (non-VSAM) data sets. CAI is the default.

CSINODE

Provide the prefix for Computer Associates common VSAM data sets (may be the same as CAINODE). CAI.VSAM is the default.

CCLISTL

Provide the name of the Computer Associates common library. This must be the same as appending CAICLIB to the prefix specified in CAINODE. CAI.CAICLIB is the default.

CLOADLIB

Provide the name of the Computer Associates common LOADLIB CAILIB. This name must be the same as appending CAILIB to the prefix you provided in CAINODE. CAI.CAILIB is the default.

CLPALIB

Provide the name of the Computer Associates common LOADLIB CAILPA. This name must be the same as appending CAILPA to the prefix you provided in CAINODE. CAI.CAILPA is the default.

CMACLIB

Provide the name of the Computer Associates common macro library. This name must be the same as appending CAIMAC to the prefix you provided in CAINODE. CAI.CAIMAC is the default.

CPROCLIB

Provide the name of the Computer Associates common procedure library. This name must be the same as appending CAIPROC to the prefix you provided in CAINODE. CAI.CAIPROC is the default.

CISPFLIB

Provide the name of the Computer Associates common ISPF message library. This name must be the same as appending CAIISPM to the prefix you provided in CAINODE. CAI.CAIISPM is the default.

CINSTOP

Provide the name of the Computer Associates common installation options library. This name must be the same as appending PPOPTION to the prefix you provided in CAINODE. CAI.PPOPTION is the default.

CSRCLIB

Provide the name of the Computer Associates common source library. This name must be the same as appending CAISRC to the prefix you provided in CAINODE. CAI.CAISRC is the default.

3.1.3 CA-ASM2 Data Set Prefixes

RELPFX

Provide the prefix to be used for CA-ASM2 version dependent data sets (such as an SMP/E DLIB). Data sets with this prefix are expected to be superseded with each new version. (There are only non-VSAM version dependent data sets.) The prefix may be up to 21 characters long. CAI.ASM2.R42 is the default.

PREFIX

Provide the prefix to be used for CA-ASM2 non-VSAM data sets. Data sets with this prefix are not expected to be superseded with each new release. The prefix may be up to 21 characters long. CAI.ASM2 is the default.

VPREFIX

Provide the prefix to be used for CA-ASM2 VSAM data sets. Data sets with this prefix are not expected to be superseded with each new release. The prefix may be up to 21 characters long. CAI.ASM2V is the default.

3.1.4 SMP/E Zone Names

CAITGT

Provide the name for Computer Associates common product SMP/E target zone. There can be a maximum of seven characters. CAITGT is the default.

CAIDLIB

Provide the name for Computer Associates common product SMP/E distribution target zone. There can be a maximum of seven characters. CAIDLIB is the default.

3.1.5 Unit and Volser Assignments

CSISER

Provide the volser for Computer Associates common SMP/E CSI (VSAM).

SMPSER

Provide the volser for all other SMP/E data sets.

SMPUNIT

Provide the unit type you specified for SMPSER above. 3390 is the default.

TGTSER

Provide the volser of the Computer Associates common SMP/E target volume.

TGTUNIT

Provide the unit type for TGTSER specified above. 3390 is the default.

DLBSER

Provide the volser of the Computer Associates common SMP/E DLIB volume.

DLBUNIT

Provide the unit type for DLBSER specified above. 3390 is the default.

WRKUNIT

Provide the esoteric name to be used for DASD work space. SYSDA is the default.

3.1.6 SMP/E Variables for USERMODS

IPOSMPE

Provide the SMP/E procedure name for your operating system. IPOSMPE is the default.

MVSTGT

Provide the target zone name for MVS. MVSTGT is the default.

MVSDLB

Provide the distribution library zone name for MVS. MVSDLB is the default.

SMPEREL

Provide the SMP/E Version number. Enter 4 for Version 1.4 or 5 for Version 1.5. The default is 4.

SOUT

Provide the SYSOUT class used in CA-ASM2 procedures and jobs. An * is the default.

3.1.7 Base Installation Parameters

TLBSER

Provide the volser of the disk volume to contain the SMP/E 'TLIB' data sets. SMP/E deletes these data sets after the product has been successfully accepted.

TLBUNIT

Provide the unit for TLBSER specified above.

PRCVOL

Provide the volser of where CAI.CAIPROC is located.

PRCUNIT

Provide the unit for PRCVOL specified above.

AVOL

This is the target volume for CA-ASM2 production data sets such as queues and work data sets. Provide the volser of where CA-ASM2 non-version dependent (which are not expected to be replaced with each version) data sets are to be put.

AUNIT

Provide the unit type for AVOL. 3390 is the default.

IPCVOL

Provide the volser of where the IPC is to reside. For integrity, the IPC and Journal should be on separate volumes.

JNLVOL

Provide the volser of where the Journal is to reside. For integrity, the IPC and Journal should be on separate volumes.

LMTYP

Select the type of information to be saved in addition to the last modification date in the DSCB. JOB specifies to save the job name or user ID. ACF specifies to save the CA-ACF2 logon ID. ACT specifies to save the first eight characters of OS accounting information. JOB is the default. Required only if the IFG0196W OPEN modification is installed.

LMOPT

Select how often the DSCB is updated with the LMTYP operand information described above for LMTYP. Select F if you want only the first update for a given day to save this information, or U if you want every update to save this information. U is the default. Required only if the IFG0196W OPEN modification is installed.

PTF196W

Provide the PTF level of IFG0196W. This specifies the current IBM PTF level (RMID) of the OPEN CSECT IFG0196W, so that CA-ASM2 can modify it. The modification is optional. It saves aging information for non-VSAM data sets that is more complete than that provided by IBM facilities. The PTF level must be specified to generate the system modifications.

PTF192C

Provide the PTF level of IDA0192C. This specifies the current IBM PTF level (RMID) of the CLOSE module IDA0192C, so that CA-ASM2 can modify it. The PTF level must be specified to generate the system modifications. If your system ages only ICF VSAM, you can eliminate the need to modify and maintain this module by replacing CSECT IDATMSTP instead. However, if you want to age VSAM clusters in VSAM catalogs, this modification is required. This modification is not required or available for DFP 3.1 and greater since the DFP level maintains all aging data.

FMIDIDAT

Provide the FMID of IDATMSTP. This is an optional method of performing VSAM aging. See PTF192C.

BASE

For installations installing the TSO Command Authorization only. Specify the hexadecimal offset in CSECT IKJEFTE2 of module IKJEFT02 at which you want the CA-ASM2 authorized TSO commands placed. You must determine this offset by examing your installation's module because this table can vary from one shop to another. The default is 18 (an empty table). TSO/E users may specify authorized TSO commands through member IKJTSOxx in SYS1.PARMLIB.

EXP

For installations installing the TSO Command Authorization only. Specify (in decimal) the amount by which the IKJEFTE2 CSECT is to be expanded. This is used if there is insufficient room remaining in the IKJEFTE2 CSECT to contain all the required entries. The default is zero. CA-ASM2 requires space for seven entries (56, 8 X 7).

PTFTE2

For installations installing the TSO Command Authorization only. Provide the PTF level (RMID) of module IKJEFTE2.

3.2 $OPTIONS

The $OPTIONS member of CAI.PARMLIB allows you to customize CA-ASM2 to operate in your unique environment. This is accomplished by specifying various fields within $OPTIONS during the installation. If you are a new user, the default values of most fields in $OPTIONS work fine during initial testing. A few fields, however, should be reviewed from the start:

- $HOMEVOL (Described on 3-10.)
- $TAPPOOL (Described on 3-13.)
- $KEEPSER (Described on 3-14.)
- $MISCOPT (Described on 3-20.)
- $SOFTWAR (Described on 3-17.)
- $MNTOPT (Described on 3-23.)
- $PREFIX (Described on 3-32.)
- $VPREFIX (Described on 3-32.)

The next topic provides an alphabetical listing of the user modifiable fields with field definitions and page references to detailed descriptions. Beginning on 3-10 is the detailed descriptions of each of the user modifiable fields.

3.2.1 Alphabetical Listing of User Modifiable Fields

Field Name	Definition	Described on
$ACSMETH	Access Method	3-19
$AMAXVER	Archive Maximum Versions	3-38
$AMINVER	Archive Minimum Versions	3-37
$ANOCPY	Archive No Copy	3-22
$ARCHEXP	Archive Expiration	3-29
$ARCHTKC	Archive Track Copier	3-25
$ASM2FLG	ASM2 Flags	3-21
$ASM2F2	ASM2 Flag 2	3-23
$BKUPEXP	Backup Expiration	3-30
$BKUPPDM	Backup Physical Data Mover	3-26
$BKUPTKC	Backup Track Copier	3-25
$BKUTIME	Backup Time	3-16
$BMAXVER	Backup Maximum Versions	3-38
$BMINVER	Backup Minimum Versions	3-37
$BNOCPY	Backup No Copy	3-22
$CATID	Catalog ID	3-36
$CMPTYPE	Compression Type	3-40
$COMLEN	Comment Length	3-27
$COMSW	Comment Switch	3-27
$CVLEXP	Candidate Volume List Expiration	3-32
$CVLFLG	Candidate Volume List Flags	3-33

Field Name	Definition	Described on
$DFGTIME	Defrag Time	3-34
$DMERROR	Data Mover Errors	3-30
$DOWNTO	Down Threshold Option	3-41
$DPREFIX	DSA Prefix	3-40
$DSACLAS	Disk Staging Archive Class	3-39
$DSADAYS	Disk Staging Archive Days	3-39
$DSAUNIT	Disk Staging Archive Unit	3-39
$DSBCLAS	Disk Staging Backup Class	3-40
$DSBDAYS	Disk Staging Backup Days	3-40
$DSBUNIT	Disk Staging Backup Unit	3-40
$DSCBF#	DSCB Field #	3-28
$DSCBLF	DSCB Length Fields	3-28
$DSCBSW	DSCB Switch	3-27
$DUP#MAX	Duplicate Maximum	3-19
$DUPTMAX	Duplicate Tape Maximum	3-19
$HARDWAR	Hardware	3-17
$HOMEVOL	Home Volume	3-10
$INCDFLT	Incremental Default	3-26
$INSTALL	Installation Date	3-28
$INTCSG	Interface CA System Group	3-24
$ISRMLIB	CA-ASM2 Message Library	3-38
$IXROPT	IXR Options	3-33
$KEEPEXP	Keep Expiration Date	3-14
$KEEPSER	Keep Volser	3-14
$MAXCRL	Maximum Concurrent Reload	3-21
$MINSPL	Minimum Size Split	3-24
$MISCOPT	Miscellaneous Options	3-20
$MNTOPT	Maintain Option	3-23
$MSSMAX	MSS Maximum (Obsolete)	3-27
$MSSULQ	MSS Logical Request (Obsolete)	3-29
$MSSULQL	MSS Logical Request Length (Obsolete)	3-28
$M860ETI	M860 Error Tab I/O	3-35
$PDMOPT	Physical Data Mover Options	3-34
$PDSUTIL	PDS Utility	3-34
$PLMTIME	Permanent Limit Time	3-16
$PREFIX	Prefix	3-32
$PRMTIME	Permanent Time	3-15
$PRTVOL	Print Volser	3-10
$RACFOPT	RACF Options	3-35
$RELDOPT	Reload Option	3-16
$RETDAYS	Retain Days	3-37
$RETNOPT	Retention Options	3-37
$RLDGRP	Reload Group	3-38
$SASMOD	SAS Module Name	3-41
$SELDISK	Select Disk	3-12
$SMFID	SMF Record ID	3-33
$SOFTWAR	Software Options	3-17
$SYSOUTC	SYSOUT Class	3-34

Field Name	Definition	Described on
$SYSTIME	System Time	3-14
$TAPPOOL	Tape Pool	3-13
$THRESH	Threshold Default	3-41
$TPUNIT	Tape Unit	3-35
$TYPEREQ	Type Request	3-30
$UDSCB	User's DSCB	3-28
$ULMTIME	User Limit Time	3-15
$USRAUTH	User Authorization	3-20
$USRTIME	User Time	3-15
$VOLGRP	Volume Groups	3-31
$VPREFIX	VSAM Prefix	3-32
$VSAMEXM	VSAM Export Mode	3-36
$WTOSW	WTO Switch	3-11

3.2.2 Descriptions of User Modifiable Fields

The following paragraphs describe each of the fields within the $OPTIONS member of CAI.PARMLIB. In each case the default condition that CA-ASM2 was shipped with is illustrated.

Read each of the descriptions and determine whether the default condition suits your needs. If not, select your choice on the $OPTIONS Worksheet, located on A-5. You need this information during the base product installation.

If you are a current CA-ASM2 user, be sure to review all current $OPTIONS settings before doing the $OPTIONS worksheet.

3.2.2.1 $PRTVOL (Print Volser)

Default: DC BL1'00000001'

Bit	Indicates
|.... ...1|	The tape volser and file sequence of a data set in archives/backup are allowed to be displayed to users when $AI/$BI/$CI commands are issued with the VOLUME parameter specified.

3.2.2.2 $HOMEVOL (Home Volume)

Default: DC XL1'01'

This field partially controls the volume to which a data set is restored. It establishes the default criteria for target selection only and has no effect unless $SELDISK=X'01'. If effective, it is further amplified by the values of $TYPEREQ and $VOLGRP, fields also defined in $OPTIONS.

$HOMEVOL is just one of the fields in $OPTIONS which controls reload target volume selection. The setting of $SELDISK determines generally how target volume selection occurs; for example, programmatically or by using predefined

default selection criteria ($SELDISK=X'01'). If $SELDISK=X'01', then the setting of $HOMEVOL determines the relative importance of the home volume (that is, the volume from which the data set was unloaded) in selecting a reload target. If $HOMEVOL is not set to X'02', then the settings of $TYPEREQ and $VOLGRP ultimately establish the subset of available DASD which may be used as reload target volumes. However this subset is defined, the net result is that CA-ASM2 chooses the pack with the most available free space from the subset of DASD established as the reload target pool. The NEWVOL operand may be used on the $RA and $RB commands to override reload target criteria specified in $OPTIONS. If NEWVOL is specified, the reload is only attempted to the volser indicated by NEWVOL. When multivolume data set is being reloaded, these same options apply, but multiple volumes are selected from the DASD subset identified.

Bit	Indicates
XL1'02'	The volume on which the data set resided when it was archived/backed up is the only volume to which it may be restored. If this volume lacks sufficient space to contain the data set, the restore is not allowed. This setting is useful only in those relatively unusual environments in which it is crucial to reload a data set to the same volume on which it originally resided.
XL1'01'	The first choice is to try to restore to the volume on which the data set resided when it was archived/backed up. The secondary choices for the default selection criteria are established by $TYPEREQ and $VOLGRP. However, the secondary path is only examined if there is not sufficient space on the original volume to contain the data set being restored or if the original volume is not online.
XL1'00'	The default selection criteria is established by $TYPEREQ and $VOLGRP fields. The original volume is not significant in target volume selection.

3.2.2.3 $WTOSW (WTO Switch)

Default: DC BL1'00000000'

Bit	Indicates
|.... ...1|	CA-ASM2 issues a WTOR prior to performing dynamic allocation of a non-M860 tape device. There are two reasons why this may be desirable: In IXR, the WTOR minimizes the amount of time the SYSZTIOT resource is enqueued, which in turn helps keep IXR requests from backlogging. In all environments, the WTOR gives the system operator an opportunity to abort the allocation attempt. The ID of the WTOR message is $TA0001E.

3.2.2.4 $SELDISK (Select Disk)

Default: DC XL1'01'

Further qualifies the default criteria for reload target volume selection. If the setting of $SELDISK is not X'01', $HOMEVOL, $TYPEREQ, and $VOLGRP are not meaningful.

Bit	Indicates
XL1'04'	Allocation is performed using a unit name from the $RLDGRP table of $OPTIONS. The device type the data set resided on at unload time is used to look up the associated unit name to be specified during allocation. If this option is selected and the physical data mover $PDM is in use, the user should ensure that the esoteric unit names (SYSDA, and so on) associated with a specific generic unit (3350, 3390, and so on) contain only volumes of that device type, otherwise the system may select a volume for a $PDM reload that is incompatible. ($PDM must reload to a volume of the same device type the data set was unloaded from.) See also the description of $RLDGRP on 3-38. **Note:** This option should be used for SMS installations to allow SMS to select the actual reload volume. When the data set being reloaded is determined to be SMS controlled, the UNITNAME presented is ignored and SMS performs the actual volume selection. If the system ACS routines do not identify the data set as SMS controlled, the normal system allocation routines select a volume from the UNITNAME specified. In all cases, the CA-ASM2 IPC record is updated to reflect the actual volsers to which the data set is reloaded.
XL1'03'	The target volume selection is the total responsibility of the user exit, $SELVOLX. CAI.CAISRC contains a sample $SELVOLX exit which selects a suitable target volume from installation-specified generic names. The logic for doing this differs substantially between MVS and non-MVS shops. **Note that IXR uses the $IXRUSVX exit for volume selection instead of $SELVOLX.**
XL1'01'	Use the selection criteria established by the $HOMEVOL, $TYPEREQ, and $VOLGRP fields.
XL1'00'	The volume selected is a storage volume; that is, the allocation is to a nonspecific volume for a permanent data set.

3.2.2.5 $TAPPOOL (Tape Pool)

Default: DC XL1'00'

Bit	Indicates
XL1'01'	The use of the CA-ASM2 tape pool is enabled. Whenever CA-ASM2 needs a new tape, it selects a tape which is marked FREE in a tape pool data set. A tape pool data set maintains the ongoing status of CA-ASM2 tapes: FREE - available for use ARCH - archive master tape BKUP - backup master tape COPY - archive or backup duplicate tape REUS - reusable IOER - I/O error If this setting is chosen, then a preallocated pool of tapes must be set aside for the exclusive use of CA-ASM2. A tape is made available for CA-ASM2 use by creation of a record for it in a tape pool data set. The first four characters of the record must be FREE. The exact record format is described here. FREE followed by the actual volser of an available standard-labeled tape. There may be more or less tape pool data sets than those set up as the system default. By default the following tape pool data sets are established: ARCH.$TAPPOOL - initially contains volsers available for use as archive master tapes. ARCH.$DUPPOOL - initially contains volsers available for use as archive duplicate tapes. BKUP.$TAPPOOL - initially contains volsers available for use as backup master and duplicate tapes **Note:** It is crucial that no tape volser be entered in more than one tape pool data set.
XL1'00'	The use of the CA-ASM2 tape pools is disabled. All tapes are acquired from the installation's scratch tape pool. CA-ASM2's own tape management facilities do not record these tapes. Presumably, the installation prefers that CA-ASM2 select its tapes from a common, universal scratch pool, already managed by an accepted commercial tape management system. The CA-ASM2 maintenance utility, $MAINT produces a report of all tapes containing CA-ASM2 archives or backups each time it is run.

3.2.2.6 $KEEPEXP (Keep Expiration Date)

Default: DC XL1'00'

Bit	Indicates
XL1'01'	Expiration date from disk is propagated to tape data set.

Note: This option should only be set to XL1'01' when data sets are created using EXPDT. If EXPDT is not being used for data sets created on disk, setting this bit results in a tape EXPDT of ZERO for that data set.

3.2.2.7 $KEEPSER (Keep Volser)

Default: DC XL1'01'

Bit	Indicates
XL1'01'	The volser of the duplicate tape of an archive/backup master is to remain unchanged; that is, its volser is different from its master's. The external and internal volser of the duplicate tape should be the same. If a master tape becomes unusable, the customary practice is to change the CA-ASM2 catalog to point to the volser of the most current duplicate of the master for all data sets for which it formerly pointed to the master tape. This is accomplished by the $CHNGSER parameter to the ASM2MNT procedure.
XL1'00'	The volser of the duplicate tape of an archive/backup master is to be changed to the same as the master's volser. Valid only if BLP processing is specified in the TPBKUP70 step of the following procedures: ASM2EXPA, ASM2EXPP, ASM2SYSA, ASM2EXPB, ASM2SYSB, ASM2MEXA, ASM2MEXP, ASM2MSYA, ASM2MEXB, ASM2MSYB, and ASM2VOLB. This means that the internal and external volser of these tapes would be different. The only advantage would be in the event of the master becoming unusable, the duplicate being a true mirror image of the master could be mounted immediately in its place.

3.2.2.8 $SYSTIME (System Time)

Default: DC PL3'730' 2 years

Default data set retention period for system-initiated archive runs using the ASM2SYSA and ASM2DSYA procedures. $SYSTIME also supplies the default data set retention period for backup runs using the ASM2SYSB and ASM2DSYB procedures. When either the $SYSBKUP or $INCBKUP parameters are used with the ASM2SYSB or ASM2DSYB procedures, the $SYSTIME SYSIN parameter may be used to revise the longevity of the output tape.

3.2.2.9 $USRTIME (User Time)

Default: DC PL3'730' 2 years

Default data set retention period for user-initiated archive requests ($AR command). The user may override this on his $AR command by using the RETPD operand.

If ULMTIME is NOT = 0 the default data set retention period for nonpermanent user-initiated archive requests ($AR command) may be overridden by the user when his request for archival is made by using the RETPD operand of the $AR request.

3.2.2.10 $ULMTIME (User Limit Time)

Default: DC PL3'1095' 3 years

If $ULMTIME = 0 and $PLMTIME does NOT = 0 all user-initiated archive requests ($AR command) are permanent. All user-initiated archive jobs use the ASM2EXPP/ASM2DEXP procedures and the ASM2EXPA/ASM2DEXA procedures are not used.

If $ULMTIME does NOT = 0 and $PLMTIME = 0 the $ULMTIME value is the maximum retention period a user may request when using the RETPD operand of the $AR command. All user-initiated archive requests are nonpermanent (that is, the ASM2EXPP and ASM2DEXP procedures are not needed).

If $ULMTIME does NOT = 0 and $PLMTIME does NOT = 0 the $ULMTIME is the maximum user-specified retention period for nonpermanent user-initiated archive requests. Nonpermanent archives are done by the ASM2EXPA/ASM2DEXA procedures when the user's $AR command does not have the permanent keyword AND a RETPD keyword is not specified or the RETPD specified is not greater than $ULMTIME. Permanent archives are done by the ASM2EXPP and ASM2DEXP procedures when the user's $AR command has the permanent keyword or a RETPD keyword is specified which is greater than $ULMTIME. Thus the user's request is automatically segregated to the permanent tapes even though he did not specifically enter the permanent keyword. If however, you do not plan to use the CA-ASM2 permanent archiving procedures ASM2EXPP and ASM2DEXP, set $PLMTIME to PL3'0'.

3.2.2.11 $PRMTIME (Permanent Time)

Default: DC PL3'1825' 5 years

If PLMTIME = 0 $PRMTIME is not used by CA-ASM2.

If PLMTIME does NOT = 0 the $PRMTIME value is the default data set retention period for user-initiated permanent archive runs, procedures ASM2EXPP and ASM2DEXP. The user may override this when his request for archival is made by using the RETPD operand of the $AR request.

3.2.2.12 $PLMTIME (Permanent Limit Time)

Default: DC PL3'3650' 10 years

Specifies the maximum retention period a user may request when using the RETPD operand of the $AR command when asking for permanent retention.

Set $PLMTIME to PL3'0' if you do not plan to have permanent archive runs using the ASM2EXPP and ASM2DEXP procedures.

3.2.2.13 $BKUTIME (Backup Time)

Default: DC PL3'90' 90 days

Default and maximum data set retention period for user-initiated backup requests ($BK command). The user may override this on his $BK command by using the RETPD operand, but he may only revise it downwards. This field does not affect the longevity of tapes produced in $INCBKUP or $SYSBKUP runs, ASM2SYSB and ASM2DSYB procedures.

Note: The version controls provided by $AMAXVER, $BMAXVER, $AMINVER and $BMINVER all override expiration date processing; a data set may be deleted from the IPC if it has not expired, but newer versions of the same data set have made the total number go over the MAXVER limits. By the same token, even though a data set has expired, it is not deleted if deleting it would cause the number of versions of that data set to go under the MINVER limit.

3.2.2.14 $RELDOPT (Reload Option)

Default: DC BL1'10010000'

Bit	Indicates
|1... ....|	When the restore ($RA/$RB) commands specify FORCE but not NEWVOL, and the data set is already cataloged, the volume the data set is cataloged to is treated as the target volume.
|..1. ....|	The user volume selection exit $SELVOLX is not invoked.
|...1 ....|	When this bit is set, the VTOC update exit $UVEXIT is called for each volume of a multivolume data set when the data set is reloaded. If the bit is off, $UVEXIT is called for only the first volume in the multivolume set.
|.... 1...|	Specifies the version of SAS to use for unloads and/or reloads. When this bit is set, SAS Version 6 is indicated. When reset, SAS Version 5 is indicated.

Note: Most installations use the name SAS to invoke the SAS system. If the name used in your environment is different, field $SASMOD must be updated.

As of SAS version 6.07, files in 5.18 format cannot be reloaded successfully, they are copied but data is not correct. If you have SAS 5.18

data sets, you must keep a copy of SAS at some level prior to 6.07 so the data sets can be reloaded using $RA and IXR, or convert all files to the SAS 6.0x format.

3.2.2.15 $HARDWAR (Hardware)

Default: DC XL1'00'

Indicates the unique hardware your installation may wish to use in CA-ASM2 processing.

Bit	Indicates
|1... ....|	Your installation has installed a Masstor M860 and wishes to use it for any CA-ASM2 jobs. The $TPM860 label associated with the $TPUNIT field of $OPTIONS indicates the unit name to specify for the M860. (See 3-35.) The $M80ETI field of $OPTIONS must also be set for CA-ASM2 to properly identify M860 units (see 3-35).
|.1.. ....|	Set this bit on if your installation does not want to use the compaction feature of the 3480 XF tape drives. CA-ASM2 sets TRTCH=NOCOMP unless the tape's DD statement specifies either TRTCH=COMP or TRTCH=NOCOMP. To set the esoteric unit name to use for a 3480 drive with the compaction feature installed, modify the $TP348X field of $OPTIONS (see 3-35).

Caution

For those wishing to use CA-ASM2 tapes for disaster recovery, compatible hardware (IDRC capable) must be available at the recovery site to restore hardware compacted tapes.

3.2.2.16 $SOFTWAR (Software)

Default: DC BL1'00001011'

Bit	Indicates
|1... ....|	CA-ASM2 performs normal DASD volume selection at archive or backup reload time for VSAM clusters. This bit can be set safely if all VSAM clusters are cataloged in ICF catalogs and hence are not subject to volume ownership restrictions. **This bit must be set in an SMS environment**.
|.1.. ....|	The installation has CA-ACF2 installed and wishes that $AUTHEXT perform CA-ACF2 security validation.
|..1. ....|	The VTOC scan performed during archive system pass or incremental backup is initialize ICF VSAM DSCBs for conventional SU60 aging and change support. Initialization consists of changing the date-last-referenced field (DS1REFDT) to today's date and rewriting the DSCB into the VTOC. It is done only for

ICF VSAM DSCBs whose date-last-referenced field is found to contain zeros. This initialization, together with installation of the distributed exit IDATMSTP, activates conventional SU60 support for VSAM objects cataloged in an ICF catalog structure.

Note: The SU60 bit is automatically maintained for VSAM data sets if you are at DFP 2.4 or later. All MVS/SP levels maintain the SU60 bit for non-VSAM data sets.

|...1 ....| (Obsolete. Informational only.) The installation has a 3850 MSS. This bit must be on to enable archive/backup/reload to and from MSS.

|.... 1...| This bit is for installations that run incremental backups (ASM2SYSB) and wish for CA-ASM2 to control the SU60 bit setting. Turn the bit on if CA-ASM2 is to control setting the SU60 bit, otherwise turn the bit off to default to MVS SU60 control. This setting functions for both non-VSAM and VSAM data sets. The CA-ASM2 aging modifications are not necessary for this feature to be operative.

Note: The SU60 bit is automatically maintained for VSAM data sets if you are at DFP 2.4 or later. All MVS/SP levels maintain the SU60 bit for non-VSAM data sets.

|.... .1..| The installation has installed CA-Top Secret or RACF and wishes to implement the CA-ASM2 security system interface.

|.... ..1.| The queued reload facility of CA-ASM2 has been chosen by the installation. See also $MAXCRL described on 3-21).

Note: Before queued reloads for batch $RA or $RB commands are honored, $RXQEXIT must be altered to allow batch reload requests to be queued. To allow these reloads to be queued, sequence numbers 00590000 and 00600000 must be commented out. The instructions being commented out are:

```
TESTENV    #TSO       IS THIS TSO?          00590000
BNO        RETURN0    NO LET RESTORE GO     00600000
```

|.... ...1| This bit controls multiple CPU sharing to the IPC and Journal. When this bit is on, the CA-ASM2 access routines protect the catalog and Journal with a systems level RESERVE. When this bit is not on, the IPC and Journal are protected with a systems level ENQ. This bit also controls RESERVE versus ENQ processing decisions in other areas of the product.

This bit should always be set ON to preserve the integrity of the IPC and Journal on systems where multiple CPUs (multiple MVS systems) share DASD and the IPC is placed on a shared DASD volume.

3.2.2.17 $ACSMETH (Access Method)

Default: DC AL1(ISAM+SEQ+BDAM+PDSE+PDS+VSAM)

Controls which data set organizations may be archived/backed up.

Bit	Indicates
|1... ....|	Index Sequential (ISAM)
|.1.. ....|	Physical Sequential (SEQ)
|..1. ....|	Direct Access (BDAM)
|.... .1..|	Partitioned Data Set Extended (PDSE)
|.... ..1.|	Partitioned (PDS)
|.... ...1|	VSAM Clusters (VSAM)

3.2.2.18 $DUP#MAX (Duplicate Maximum)

Default: DC AL1(1)

Note: This field is valid only if $TAPPOOL is set to X'01' (see 3-13).

Represents the number of duplicate tapes of the same archive/backup master tape that are to be retained before CA-ASM2 automatically recycles the oldest of the duplicate tapes. The default of 1 ensures that the classic grandfather-father-son backup scheme is maintained. It is unwise to retain fewer duplicates. For example, to always duplicate a given master to the same duplicate tape and if the master physically breaks, the end result is that you are left without good tapes. The maximum number for this field is 9.

3.2.2.19 $DUPTMAX (Duplicate Tape Maximum)

Default: DC PL2'5'

Note: This field is valid only if $TAPPOOL is set to X'01' (see 3-13).

Represents the minimum number of days which must have passed before a duplicate tape can be reused. This provides an additional margin of safety since it may prevent reuse even though a tape is determined to be an additional duplicate according to the value of $DUP#MAX.

3.2.2.20 $USRAUTH (User Authorization)

Default: DC XL1'80'

Bit	Indicates
XL1'80'	All TSO users have the ability to do CA-ASM2 TSO restores. If the queued reload facility of CA-ASM2 has been selected, all $RA and $RB TSO reload requests are queued and $USRAUTH has no effect. If the queued reload facility of CA-ASM2 has not been selected, $RA and $RB TSO reloads involve allocating a tape drive to the TSO session. (Also see $SOFTWAR on 3-17.) This does not confer the mount attribute on a user for any purposes other than a CA-ASM2 retrieval. Also, it does not allow mounting of a disk to serve as a reload target. It only allows mounting of the archive or backup tape from TSO.
XL1'00'	The ability to do TSO restores is limited to those users who have the mount attribute as defined in their profile table in SYS1.UADS.

3.2.2.21 $MISCOPT (Miscellaneous Options)

Default: DC BL1'11001100'

Bit	Indicates
|1... ....|	The CA-ASM2 PDS data mover is allowed to unload data to tape blocked at 32K. This is ideal for performance in moving the data. However, if this option is chosen, PDSs unloaded by CA-ASM2 are not reloadable by IBM utilities. They may still be reloaded using an independent CA-ASM2 utility ($PDSUR). It is independent in the sense that it can reload a PDS it has unloaded at 32K from an independent job whether or not a CA-ASM2 catalog entry exists for the data set.
|.1.. ....|	$SM does not uncatalog data sets not found on volumes to which they are cataloged. To further clarify, if this bit is off, then $SM uncatalogs data sets not located on the volume to which its system catalog entry points. This is a handy catalog cleanup tool unless there is dependence in your installation on techniques such as VOL=ref to effectively use a single catalog entry to point to several data sets. If such is the case, disable the catalog cleanup facility in $SM by turning this bit on.
|..1. ....|	TSO/E environment. Currently this only affects the method in which CA-ASM2 commands are invoked by CA-ASM2 through the ISPF dialog manager. $SM is invoked through the TSOEXEC service. **This bit is not applicable for versions 4.2 and above of CA-ASM2.**
|...1 ....|	System-pass (incremental) backup of ICF user catalogs is requested. Backup occurs unconditionally without regard to change status whenever an ICF user catalog is encountered on a subject volume.

|.... 1...|	The CA-ASM2 deferred-scratch facility is enabled for archive disk-to-tape unload runs. All archive scratch processing of unloaded data sets occur in the TPBKUP70 step only.
|.... .1..|	Allows user-initiated archival of VSAM clusters that have alternate index associations. If this bit is not set, $AR requests for VSAM clusters with alternate index associations are denied. For deferred-scratch in disk-to-disk archive unload runs, this option must be enabled along with the $ANOCPY option enabled (Y), and the $DEFRSCR parameter specified to both $COPYTP and $DASDMNT to override the default immediate scratch. This bit is only referenced for unload requests for VSAM spheres ($AR). If a user issues a $AR for a VSAM sphere and this bit is not set, the $AR request is rejected. This bit is not significant for reloads, either implicit (IXR) or explicit ($RA/$RB).
|.... ..1.|	Disables the in-core generation of Format-1 DSCBs during incremental backup runs. Setting this bit considerably reduces virtual storage requirements at the expense of substantial performance improvement.
|.... ...1|	Flags the CA-ASM2 catalog record for deletion following a successful restore/reload operation. **This field is only applicable for a reload from an archive.**

3.2.2.22 $ASM2FLG (ASM2 Flags)

Default: DC BL1'00000000'

Bit	Indicates
|...1 ....|	The $ENDULOD exit is enabled. The End-of-Unload exit is described in Chapter 4, “CA-ASM2 Exit Descriptions” on page 4-1 page=no,.
|.... .1..|	Reloads are allowed to offline real DASD.
|.... ..1.|	Unloaded tape data sets are password protected if the disk data set is password protected. Use with caution, as this causes S913-30 abends if nonprotected and password protected data sets are intermixed during an unload run.

3.2.2.23 $MAXCRL (Maximum Concurrent Reload)

Default: DC AL1(5)

Represents the maximum number of concurrent TSO reload operations allowed per system. Its purpose is to limit the number of tape drives used at the same time for TSO reloads. If $MAXCRL is set to DC AL1(0) and the queued reload facility of CA-ASM2 has not been chosen (see $SOFTWARE on 3-17), then users can only perform $RA/$RB reloads by executing them through a batch job.

3.2.2.24 $ANOCPY/$BNOCPY (Archive/Backup No Copy)

Default: DC BL1'00000001'

Bit	Indicates
|.... .1..|	Indicates to suppress duplexing the current volume in a backup run, thus only duplexing backup tapes after they fill up.
|.... ..1.|	Indicates that archive runs do not make a tape copy. The bookkeeping performed in ARCH.LOxxxx data sets necessary to remember which tapes to duplicate is not performed unless deferred scratch processing is used. If this bit is on and deferred scratch is not specified in $MISCOPT, the TPBKUP70 step of archive procedures could be deleted (commented out), or jobs using the archive procedures (ASM2EXPA, ASM2EXPP, ASM2SYSA, ASM2MEXA, ASM2MEXP, ASM2MSYA) could have overriding statements

```
//TPBKUP70.INTAPE DD DUMMY
//TPBKUP70.OUTAPE DD DUMMY
```

to execute the TPBKUP70 step without allocating tape drives.

|.... ...1|	Indicates that backup runs do not make a tape copy. The bookkeeping performed in BKUP.LOxxxx data sets necessary to remember which tapes to duplicate is not performed. The TPBKUP70 step of backup procedures could be deleted (commented out), or jobs using the backup procedures (ASM2EXPB, ASM2SYSB, ASM2VOLB ASM2MEXB, ASM2MSYB) could have overriding statements

```
//TPBKUP70.INTAPE DD DUMMY
//TPBKUP70.OUTAPE DD DUMMY
```

to execute the TPBKUP70 step without allocating tape drives.

3.2.2.25 $ASM2F2 (ASM2 Flag 2)

Default: DC XL1'00'

Bit	Indicates
|.... ...1|	Generation of an extended File 1 of all CA-ASM2 master tapes is disabled. The extended File 1 takes up 5 feet of tape on a 3480, 37 feet on a 3420. If this bit is set, CA-ASM2 cannot ensure that duplex tapes are large enough to contain full copies of CA-ASM2 master tapes. 3490s appear the same as 3480s to MVS and CA-ASM2.

3.2.2.26 $MNTOPT (Maintain Option)

Default: DC BL1'00000001'

Bit	Indicates
|.... ...1|	Setting this bit results in archive tapes automatically being recycled through the IPC and CA-1. Duplicate tapes are determined and expired as explained for |.... ...0| next. This method narrows down the amount of manual updates but if the output (list of tapes with expired data sets) from the archive are lost, the same tapes are not listed in the next archive run resulting in CA-1 holding the duplex tapes with EXPDT=99365 or EXPDT=99001 which only holds the current cycle. If this occurs, execute the TMSGRW utility of CA-1 to produce a list of master and duplicate tapes. See Tape Management Support in *CA-ASM2 Getting Started* for further information.
|.... ...0|	This setting results in the archive tapes not being recycled through CA-ASM2. This requires a manual update using the output generated from the archive run to determine when tapes should be expired. To accomplish this, use ASM2MNT with the $PURGETP control statement to expire the tape from the IPC and CA-1. The duplicated tapes would also have to be manually recycled. To determine this, set $DUPEDSN SETC 'YES' in $XTTMS. This causes the final qualifier of the duplicated tape to be Txxxxxx where xxxxxx indicates the master tape volser. This volser on the dsn is used as a tool to tie a duplicated tape back to the master tape that shows up in the output of the archive run. You would then follow the same steps to recycle the duplicated tapes. If the output from the archive run is lost and the tapes do not get recycled, the same tapes come out again on the next archive run. Using this method keeps both the IPC and CA-1 in sync with each other. The tapes should not be manually expired through CA-1 since there is not a method in place for CA-1 to talk to CA-ASM2. This results in the IPC containing invalid entries for data sets that no longer reside on a tape, causing the IPC and CA-1 to be out of sync.

Note: $MNTOPT works the same way for TLMS as for CA-1 described above except that TMSGRW is only available with CA-1. For maintaining duplex tapes automatically, see $RSUCA1 for CA-1 Version 4.x, $RSUCA15 for CA-1 Version 5.1 or greater, or $RSUTLMS for TLMS. More information is under the topic Tape Management Support in *CA-ASM2 Getting Started*.

|.... ..1.| Overrides normal handling of backup records logically deleted during $MAINT processing. When this bit is on, once criteria for deletion have been met (retention period, $BMINVER, $BMAXVER), backup records are removed from the IPC rather than just marked as being logically deleted.

|.... ..0.| This is the default. No unload record is physically deleted from the IPC until all records for that tape volume have expired and the $RETDAYS time interval has passed.

3.2.2.27 $INTCSG (Interface Cpmputer Associates System Group)

Default: DC BL1'00000000'

Bit	Indicates
|.1.. ....|	Allow DISP=SHR access during backup to VSAM clusters that cannot be accessed with DISP=OLD. This applies only to clusters defined without a 4 for either SHAREOPTION: SHR(4) clusters residing on shared DASD are always allocated with DISP=SHR. Other clusters, including SHR(4) clusters on nonshared DASD, are normally allocated DISP=OLD during backup for integrity reasons and thus cannot be backed up when they are allocated to another memory. This bit allows them to be backed up with DISP=SHR with the stipulation that integrity of the backup is not guaranteed if the other allocation is updating.
|..1. ....|	Generate deferred scratch verbose diagnostics.
|.... 1...|	Do not fail the archive of a VSAM sphere if an archive of an associated alternate index fails.
|.... .1..|	Do not include VSAM clusters in the backup of the VTOC taken during incremental backup runs.
|.... ...1|	$VGENDSP gets the VSAM master password. This is used to unload password-protected VSAM clusters.

3.2.2.28 $MINSPL (Minimum Size Split)

Default: DC AL2(100)

Represents the minimum size data set (in tracks) that is eligible to be split across archive/backup tapes. A specification of 9999 ensures that no data sets are split across multiple tapes.

Note: If 9999 is specified and a data set cannot fit on a tape, the unload is failed after trying to unload it as the only data set on a tape.

3.2.2.29 $ARCHTKC (Archive Track Copier)

Default: DC BL1'00000000'

Applicable to archive runs only. Describes which data set organizations is handled by the CA-ASM2 physical track copier ($PDM). Since this data mover merely produces a track image, there are no restrictions as to what it copies. The data set must be restored to the same device type from which it was unloaded.

Note: CA-ASM2 $PDM must be available at disaster site to reload these data sets.

Unmovable data sets must be preallocated to their original track location.

The intention is that this data mover is invoked only when it complements the capabilities of the CA-ASM2 logical data movers (some restrictions on what is unloaded, but typically no restrictions on reload operation). CA-ASM2 facilities are provided to preallocate the target disk data set for the user in either case.

This is not recommended for archiving. The possibility exists that the device type a data set was unloaded from may not remain in the data center, making it impossible to reload the data set.

Bit	Indicates
|..1. ....|	Unmovable.
|...1 ....|	DA data sets without LSTAR pointer.
|.... 1...|	Direct, (BDAM) relative or absolute track.
|.... .1..|	Bad attributes (invalid DSCB).

3.2.2.30 $BKUPTKC (Backup Track Copier)

Default: DC XL1'00'

Applicable to backup runs only. Describes which data set organizations is handled by the CA-ASM2 physical track data mover ($PDM). As noted in the description of $ARCHTKC, physical data movers typically do not provide device independence. This can be a serious problem when archiving since a couple of years may pass before a data set is dearchived. It is much less a problem for data set backups since restoration generally occurs much sooner after the data is backed up. On the positive side, the physical track data mover has the advantage of being a faster, more efficient data mover.

Bit	Indicates
|1... ....|	SYSCTLG
|..1. ....|	Unmovable
|...1 ....|	DA data set without LSTAR pointer

|.... 1...|	Direct (BDAM) relative or absolute track
|.... .1..|	Bad attributes (invalid DSCB)

3.2.3 $BKUPPDM (Backup Physical Data Mover)

Default: DC XL1'00'

Applicable to backup runs only. Describes which data set organizations is handled by the CA-ASM2 physical track data mover ($PDM) even though a logical data mover could process the data set. The advantage of this data mover is speed. The disadvantage is that there are unwanted but necessary restrictions on restorability; for example, the data set must be restored to the same device type from which it was unloaded.

Unmovable data sets, including ISAM data sets, must be preallocated to their original track location.

The intention is that this data mover is invoked when it complements the capabilities of the CA-ASM2 logical data movers (some restrictions on what is unloaded, but typically no restrictions on reload operation). CA-ASM2 facilities are provided to preallocate the target disk data set for the user in either case.

Bit	Indicates
|1... ....|	Sequential data sets
|.1.. ....|	Partitioned data sets (PDSs)
|..1. ....|	Direct (BDAM) relative record data sets
|...1 ....|	ISAM
|.... 1...|	Empty data sets
|.... .1..|	Partitioned Data Set Extended (PDSEs)

3.2.3.1 $INCDFLT (Incremental Default)

Default: DC PL2'1'

Applicable to $INCBKUP runs only. Represents the default number of days since the last $INCBKUP run if no operands were included on the $INCBKUP SYSIN parameter and the volume has never before been through $INCBKUP.

Using the default value of 1, if the first incremental backup of pack DASD01 is run at 12:33 PM on August 24th, then going back 1 day means the first incremental backup uses 00:00 AM of August 23rd as its default base date; that is, it unloads only data sets created or changed since that time. In the next incremental backup run of DASD01, CA-ASM2 would use 12:33 PM of August 24th as its base date.

3.2.3.2 $MSSMAX (Mass Storage Maximum)

Default: DC AL1(00)

(Obsolete. Informational only.) Field effective for MSS users only. Defines the maximum number of virtual volumes which a CA-ASM2 run is ever to allocate (stage) concurrently. This must be less than the number of virtual UCBs generated for MSS use: consider the value carefully as a high value is good for CA-ASM2 performance but impacts your other users. (As a guideline only, about one-quarter of your defined MSS UCBs.) A value of 3 or less is definitely not recommended.

Used by the $MIGRATE function when the MSSO parameter is used.

3.2.3.3 $COMSW (Comment Switch)

Default: EQU 0

0 = No user comments to be carried in the catalog record.

1 = User comments are to be carried in the catalog record.

3.2.3.4 $COMLEN (Comment Length)

Default: EQU 30

This field sets the maximum length of the comment that can be carried in the catalog record. Cannot be specified unless $COMSW above is set to 1 (maximum 30 bytes).

Note: If $COMSW = 1 and $COMLEN does not equal 0, the COM keyword of the $AR/$BK commands is operative.

3.2.3.5 $DSCBSW (DSCB Switch)

Default: EQU 0

0 = no DSCB fields are to be saved in the catalog record for the restoration at reload time. When set to 0 the $DSCB#, $DSCBLF and $UDSCB fields are ignored.

1 = DSCB fields are to be saved in the catalog record and restored upon successful reloads. The $DSCBF#, $DSCBLF and $UDSCB fields describe which F1 DSCB fields are to be saved and restored after reload.

Note: Starting with Version 4, a copy of the entire DSCB is kept in the IPC. This field and the three following related fields only pertain to restoring DSCB fields after successful reloads.

3.2.3.6 $DSCBF# (DSCB Field #)

Default: EQU 0

Defines the number of noncontiguous DSCB fields to be saved (limit of 10). See also the description of the $DSCBSW, $DSCBLF and $UDSCB fields.

Note: If $DSCBSW = 1, $DSCBF# can specify up to 10 data areas in a DSCB to be preserved across archival or backup of the data set; that is, the data areas in the reloaded data set's DSCB are identical to what they were when the data set was unloaded. Consider the value of this option if accounting information or installation-specific security data are maintained in the DSCB.

3.2.3.7 $DSCBLF (DSCB Length Fields)

Default: EQU 0

Defines the total length of all DSCB fields that are to be preserved in the catalog record. See also the description of the $DSCBSW, $DSCBF# and $UDSCB fields.

3.2.3.8 $UDSCB (User's DSCB)

Default: DS 0X

For each DSCB field to be preserved, place two DCs defining offset in the DSCB and its length. For example, to retain the expiration date field of the DSCB, DCs would look like this:

```
DC     AL1(12)     12 bytes from the F1 format ID
DC     AL1(3)      3 bytes in length
```

Limit of 10 pairs. See also the description of the $DSCBSW, $DSCBF# and $DSCBLF fields.

3.2.3.9 $INSTALL (Installation Date)

Default: DC CL5'00000'

Enter the date you installed CA-ASM2 here in the form yyddd. (98033 indicates February 2, 1998.) This data is not used by the system, it is for reference only.

3.2.3.10 $MSSULQL (Mass Storage System Logical Request Length)

Default: DC AL1(0)

(Obsolete. Informational only.) Set to the number of significant characters in the $MSSULQ field (following).

3.2.3.11 $MSSULQ (Mass Storage System Logical Request)

Default: DC CL9' '

(Obsolete. Informational only.) From one to nine characters defining the prefix for MSS-only logical request queues. Prefix in this context means leading generic character string. For example, if the specified prefix is 'm', then any $AR or $BK requests to queues beginning with the character 'm' are treated as unloadable in MSS-only runs causing CA-ASM2 to optimize its processing techniques for MSS. A match to the value specified for this field may appear as the operand of the QNAME keyword of $AR/$BK. Implies special tapes are used for unloading MSS data to tape. Users may direct data to any logical queue they wish with the result that data targeted for similarly named queues are unloaded to common tapes.

3.2.3.12 $ARCHEXP (Archive Expiration)

Default: DC AL1(99) year of expiration
DC AL2(365) day of expiration

Specifies expiration date to use for archive tapes; only file 1 on the tape is affected.

Files 2-n are always given a retention period = 0. To understand the importance of this, consider the situation in which a disk I/O error occurs dumping file 37 to tape. CA-ASM2 logically attempts to continue processing with the next data set which it also attempts to write to file 37. If the first data set were expiration date protected, then the second data set would get a 713 in a CA-1 environment or require a correct operator reply in other environments. Forcing RETPD=0 in all cases avoids these problems.

The reason for the default setting of 99365 is to minimize the likelihood of accidental scratching of CA-ASM2 archive master tapes. CA-ASM2 purposely avoids automatic recycling of its archive masters. Instead it provides constant reminders of how many unexpired data sets exist on each tape. Presumably, users are notified shortly before all data sets on a tape expire that the entire reel is to be recycled soon. This eleventh hour warning is a last opportunity to reload anything they want from the tape. A tape is said to be expired in the CA-ASM2 sense when all data sets on the tape have zero or negative days left until expiration according to the archive catalog. The expiration date on the tape label itself is not meaningful to CA-ASM2.

$ARCHEXP may be overridden in the JCL. Therefore, different archive procedures may have different tape retention periods. If a commercial tape management system such as CA-Dynam/TLMS or CA-1 is present, and the installation wants the system to control archive tape recycling, then a RETPD=nnn JCL specification is appropriate.

If CA-1 is present and for some reason archive tapes are not under its control, 98000 should be entered in this field.

3.2.3.13 $BKUPEXP (Backup Expiration)

Default: DC AL1(00) year of expiration
DC AL2(000) day of expiration

Specifies expiration date to use for backup tapes; Only file 1 of the tape is affected.

Files 2-n are always given a retention period = 0.

Note: If CA-ASM2 tapes are to be under the control of a tape management system, but CA-ASM2 is to expire those tapes, then $BKUPEXP should be set to 99365.

The reason for the default setting of RETPD=0 is that in all environments this makes it easy for CA-ASM2 to automatically reuse one of its backup tapes. In general, there is no reason to expect that anyone has second thoughts about recycling backup tapes.

$BKUPEXP may be overridden in the JCL. Therefore, different backup procedures may have different tape retention periods. If a commercial tape management system such as CA-Dynam/TLMS or CA-1 is present, and the installation wants the system to control backup tape recycling, then a RETPD=nnn JCL specification is appropriate.

If CA-1 is present and for some reason backup tapes are not under its control, then label=EXPDT=98000 should be specified.

3.2.3.14 $DMERROR (Data Mover Errors)

Default: DC AL2(9999)

Represents the maximum number of consecutive data mover failures allowed. When the maximum number of failures is reached the archive/backup run terminates with return code 36. Setting $DMERROR to 9999 indicates that there is no limit on the number of consecutive data mover failures allowed.

Also applies to restores. Restores that normally would terminate the run are now counted, and restores continue until the count equals this value.

During disk-to-disk unloads if three or more allocation errors are encountered, the job terminates. The value in $DMERROR is not considered in such a situation.

3.2.3.15 $TYPEREQ (Type Request)

Default: DC CL1'2'

$TYPEREQ and $VOLGRP are ignored unless $SELDISK = X'01'. (See 3-12 for information on $SELDISK.) When used, $VOLGRP defines a subset of available online DASD from which CA-ASM2 selects as a reload target candidate the volser with the most available free space. If $HOMEVOL = X'00' then this is the primary and only path from which a reload target is selected. If $HOMEVOL = X'01' then this is the secondary selection path which is used only if there is not sufficient free space available to restore a data set to its home volume. (See 3-10

for further information on the $HOMEVOL field.) For multivolume data sets, the same rules apply, except as many volumes as necessary (up to the number the data set originally resided on) are selected from the group identified by $TYPEREQ/$VOLGRP.

$TYPEREQ and $VOLGRP are also ignored when the batch user simply selects his own reload target volser using the NEWVOL operand of the $RA and $RB commands. The value of $TYPEREQ (1, 2, 3, 4, or 5) defines the meaning of the $VOLGRP fields which follow.

Caution

SMS users must not code SMS controlled volumes in $VOLGRP either explicitly or through a generic specification.

Value	$VOLGRP field
0	Indicates that any storage volume may be used.
1	Indicates that $VOLGRP consists of a list of volsers (six characters long including trailing blanks) terminated by a value of X'00'.
2	Indicates that $VOLGRP defines one or more generic sets of DASD volsers specified by successive field pairs. The first field in each pair defines the length of the second field. The pair sets are terminated with an X'00' byte which indicates a zero length field follows. All volsers entered in $VOLGRP for $TYPREQ=2 must be online volsers.
3	Indicates that $VOLGRP contains a single, variable-length character string whose length is specified by the first byte at $VOLGRP. Suitable volsers are any volsers whose ddname is specified in the TSO logon procedure or batch JCL begins with the specified character string.
4	Indicates that $VOLGRP contains a single, variable-length character string whose length is specified by the first byte at $VOLGRP. Suitable volsers are any volser whose ddname as specified in the TSO startup procedure begins with specified character string.

3.2.3.16 $VOLGRP (Volser Group)

Be sure to read the description of $TYPEREQ immediately preceding.

Note: A maximum of 79 bytes may follow.

Default: DC X'03'
DC C'TSO'
DC X'04'
DC C'DATA'

The above is an example only. It applies to a shop for which volumes suitable for use as reload target volumes have volsers which begin 'TSO' or 'DATA'. In this particular example, CA-ASM2 selects the online DASD volume whose volser

begins with either 'TSO' or 'DATA' which has the most free space and use it as the reload target volume.

Caution

SMS users must not code SMS controlled volumes in $VOLGRP either explicitly or through a generic specification.

3.2.3.17 $PREFIX (Prefix)

Default: DC C'CAI.ASM2'

Enter in the above line a DC C'xxxx' where xxxx is the CA-ASM2 non-VSAM data set prefix. Do not include trailing period. The maximum size of the CA-ASM2 prefix is 21 characters. Do not specify a length in above DC.

This option specifies the prefix for ALL non-VSAM data sets. If you do not want a prefix, change C'CAI.ASM2' to C' '. This option enables you to select the high-level indexes associated with CA-ASM2 non-VSAM data sets and for new release versions of CA-ASM2 to immediately establish a test environment consisting of strictly nonproduction CA-ASM2 data sets.

For VSAM data set prefixes, see $VPREFIX.

3.2.3.18 $VPREFIX (VSAM Prefix)

Default: DC C'CAI.ASM2V'

Enter in the above line a DC C'xxxx' where xxxx is the CA-ASM2 VSAM data set prefix. Do not include trailing period. The maximum size of the CA-ASM2 prefix is 21 characters. Do not specify a length in above DC.

This option specifies the prefix for ALL VSAM data sets. If you do not want a prefix, change C'CAI.ASM2V' to C' '. This option enables you to select the high-level indexes associated with CA-ASM2 VSAM data sets and for new release versions of CA-ASM2 to immediately establish a test environment consisting of strictly nonproduction CA-ASM2 data sets.

For non-VSAM prefixes, see $PREFIX above.

For IPC and Journal data set names, see also $CATID (Catalog ID) on 3-36.

3.2.3.19 $CVLEXP (Candidate Volume List Expiration)

Default: DC PL4'4'

This field is for IXR/JES3 shops only. It represents the number of days a candidate volume list should be considered active. When a job enters the JES3 converter, a LOCATE is issued for all cataloged data sets. IXR intercepts the LOCATE request for all uncataloged data sets or data sets that are cataloged to a fictitious volser and assigns a candidate volume list to the data set. The candidate list is returned to JES3 from LOCATE; JES3 uses the list for DSN ENQ and set

up processing. IXR updates the IPC with the volume list and at restore uses the candidate list for volume selection.

3.2.3.20 $CVLFLG (Candidate Volume List Flags)

Default: DC BL1'00000000'

This field is for IXR/JES3 shops only.
Indicates how the home volume should be used in candidate volume selection.

Bit	Indicates
|1... ....|	Put the home volume as the first volume in the candidate volume list. Then select four additional volumes from the CVOL list to complete the list of candidate volumes to return to JES3.
|.1.. ....|	The HOMEVOL was used to build the candidate volume list and should be the primary volume for reload volume selection (that is, $HOMEVOL, then $VOLGRP). Note: This bit is valid only if the |1... ....| bit is on.
|..1. ....|	Only use home volume if it is online to this system.

3.2.3.21 $SMFID (SMF Record ID)

Default: DC AL1(0)

If you are using IXR and want to produce SMF records for billing reloads, specify the SMF record ID to use. The format of the record can be found in CAI.CAIMAC(#IXRSMF).

3.2.3.22 $IXROPT (IXR Options)

Default: DC BL1'00000001'

Bit	Indicates
|.... ...1|	Use the deferred option to allocate tape drives for IXR. This stops subsystem lockout which occurs because dynamic allocation issues an ENQ on the SYSZTIOT resource. IXR issues a message ($TA0455D) to tell the operator to mount the required tape. When the tape is mounted, a device end interrupt is generated and flagged in the UCB. IXR waits until the UCB is flagged before it allows the reload task to continue. MVS/SP 1.3.2 and higher supports dynamic allocation's DEFER mount function. If the MVS system does not support the function, turn this option off.

Note: $IXROPT activates a feature where IXR uses deferred mounting of tapes when envoking the MVS dynamic allocation service, SVC 99. If this bit is not on, (using MVS mount messages instead) there is a significant performance impact on IXR reload analysis and reload I/O throughput. This bit being turned on is the preferred method of running IXR.

3.2.3.23 SYSOUTC (SYSOUT Class)

Default: DC X'00'

This is the SYSOUT class used to dynamically allocate AMSOUT and SYSPRINT for IXR. If you specify X'00', you allocate to the default SYSOUT class for the IXR address space. If you want to specify a specific class; that is, for CLASS=A you would change the above field to DC C'A'.

3.2.3.24 $PDMOPT (Physical Data Mover Options)

Default: DC B'10000001'

Indicates the CA-ASM2 physical data mover ($PDM) processing options.

Bit	Indicates
|1... ....|	The CA-ASM2 physical disk read program utilizes the RMCKD (read multiple-count-key-data) channel command to read an entire physical disk track with one channel program. This is a performance enhancement item for unloads that are done by $PDMU or $DEFRAG. The results of the read are verified and if anything appears wrong, the track is reread with a read count followed by a read count-key-data if necessary.
|.... ...1|	EXCPVR is not used by the CA-ASM2 physical disk read program during archives, backups, or full volume unloads.

3.2.3.25 $PDSUTIL (PDS Utility)

Default: DC CL8'IEBCOPY'

The name of the program to be used for $SM compress. IEBCOPY requires authorization, and may not be used under SPF. Users that have SPFCOPY should replace IEBCOPY with SPFCOPY.

3.2.3.26 $DFGTIME (Defrag Time)

Default: DC AL2(30) 30 days

Default data set retention period for tapes produced by $DEFRAG. User requests can be revised downward by using the RETPD operand of the 'SYSUT2' JCL statement.

3.2.3.27 $TPUNIT (Tape Unit)

```
Default: $TP3420 DC CL8 'TAPE    ' unit name for 3420 allocations
         $TP3480 DC CL8 '3480    ' unit name for 3480 allocations
         $TPM860 DC CL8 'M860    ' unit name for M860 allocations
         $TP348X DC CL8 '3480X   ' unit name for 3480 XF drives
                 DC CL8 '        ' unit name table expansion
                 DC CL8 '        ' unit name table expansion
                 DC CL8 '        ' unit name table expansion
```

You should change the above unit names to conform to your installation standards. **Important**: Do not delete or reorder any of the above Define Constants.

Note: 3490s appear as 3480s to MVS and CA-ASM2.

3.2.3.28 $M860ETI (M860 Error Tab I/O)

Default: DC X'00'

You should change the above field to the value that was specified in the ERRTAB= parameter of the IODEVICE macro during SYSGEN for the M860 UCBs. This is used to determine if an unload device is an M860 or a normal tape drive.

3.2.3.29 $RACFOPT (RACF/CA-Top Secret Options)

Default: DC BL1'00000000'

Bit	Indicates
|.... ..1.|	Override ADSP for reloads. (RACF only.) Tells CA-ASM2 to use the CA-ASM2 profile saved at unload time instead of the RACF profile created by the ADSP option. If CA-ASM2 determines that the reloading user ID has the ADSP option (and that the profile was built by ADSP) CA-ASM2 deletes the ADSP profile and restore the profile saved by CA-ASM2 when the data set was unloaded.
|.... ...1|	Always-Call. (CA-Top Secret or RACF.) Allows CA-ASM2 to support environments that have Always-Call.

Note: $SOFTWAR must have the X'04' bit turned on in order to activate the security interface. (See $SOFTWAR description on 3-17.)

3.2.3.30 $VSAMEXM (VSAM Export Mode)

Default: DC BL1'00000000'

Bit	Indicates
|1... ....|	Use CIMODE for VSAM ESDS backup export
|..1. ....|	Use CIMODE for VSAM ESDS archive export

CA-ASM2 normally archives or backs up VSAM data sets by invoking IDCAMS EXPORT processing in RECORDMODE. This option controls whether IDCAMS EXPORT is performed in RECORDMODE or CIMODE for VSAM ESDS organizations only. VSAM KSDS and RRDS organizations may only be performed in RECORDMODE, while the new LDS (VSAM Linear Data Set) organization may only be EXPORTed in CIMODE. To enable CA-ASM2 to backup or archive DB2 databases, you must set these bits on.

If CIMODE is used during EXPORT processing, the data set can only be reloaded (IMPORTed) on a system that supports CIMODE.

Note: There are no other specific bits for linear data sets. Since linear data sets may only be exported in CIMODE, CA-ASM2 only does it this way.

3.2.3.31 $CATID (Catalog ID)

Default: DC C'ASM2'

Defines the Integrated Product Catalog's ID. This field is used to construct the ddnames and the DSNs for the IPC and IPC Journal data sets for dynamic allocation. $CATID may be from one to four characters in length. It serves as a ddname prefix with IPC and JNL concatenated as the suffix. If $CATID was specified as ASM2, ASM2IPC and ASM2JNL would be the ddnames constructed. The DSNs are constructed using $VPREFIX as the data set name prefix with the ddname as the last node. If $CATID is specified as ASM2 and $VPREFIX specified as SYS2, the IPC and Journal would be allocated as:

```
//ASM2IPC  DD  DSN=SYS2.ASM2IPC,DISP=SHR
//ASM2JNL  DD  DSN=SYS2.ASM2JNL,DISP=SHR
```

CA-ASM2 always dynamically allocates the IPC and Journal cluster if not already allocated through JCL. Therefore, once the IPC and Journal cluster names and associated ddnames are defined through $CATID in $OPTIONS, coding IPC and Journal DD statements in any CA-ASM2 job is optional.

3.2.3.32 $RETNOPT (Retention Option)

Default: DC X'00'

Bit	Indicates
|.... ...1|	Specifies the retention period for the $UA, $UB and $CU commands is to be based on today's date. If not specified (default), the unload date is used.

Bit	Indicates
|.... ..1.|	Specifies the value in $PLMTIME is used to determine the retention of data sets unloaded by $DASDMNT if the data set has an expiration date of 99365 in the F1 DSCB.

3.2.3.33 $RETDAYS (Retain Days)

Default: DC A(7)

Defines the number of days after retention expiration that an archive or backup is retained in a logically deleted status. After this period expires, the data set is purged from the IPC by the $MAINT utility when all data sets on the tape are in logically deleted status. Until then, the archive or backup may be undeleted with the $CU command with the UNDELETE parameter.

3.2.3.34 $AMINVER (Archive Minimum Versions)

Default: DC AL2(0)

Specifies the minimum number of versions of archived data sets that CA-ASM2 is to maintain. This minimum number of archive copies of each data set are not automatically deleted by the maintenance program $MAINT, even though they may have expired based on their retention periods (regardless of their expiration date). Explicit requests to delete data sets through the $DA, $DB, $CU and ISPF DELETE commands do not honor this minimum limit.

These version limits are enforced only if the recycling of archive tapes is enabled through the $MNTOPT field of $OPTIONS.

The version limit is applied to all data sets maintained by CA-ASM2. This version control parameter overrides expiration date retention processing.

3.2.3.35 $BMINVER (Backup Minimum Versions)

Default: DC AL2(0)

Specifies the minimum number of versions of backed up data sets that CA-ASM2 is to maintain. This minimum number of backup copies of each data set is not automatically deleted by the maintenance program $MAINT, even though they may have expired based on their retention periods (even if the data set has expired).

3.2.3.36 $AMAXVER (Archive Maximum Versions)

Default: DC AL2(99)

Specifies the maximum number of versions of archived data sets that CA-ASM2 is to maintain. CA-ASM2 automatically selects the oldest versions above this limit for deletion by the maintenance program $MAINT, even though those older versions may not have expired based on their retention periods.

These version limits are enforced only if the recycling of archive tapes is enabled through the $MNTOPT field of $OPTIONS.

This version control parameter overridesexpiration date retention processing.

3.2.3.37 $BMAXVER (Backup Maximum Versions)

Default: DC AL2(3)

Specifies the maximum number of versions of backed up data sets that CA-ASM2 is to maintain. CA-ASM2 automatically selects the oldest versions above this limit for deletion by the maintenance program $MAINT, even though those older versions may not have expired based on their retention periods.

This version control parameter overrides expiration date retention processing.

3.2.3.38 $ISRMLIB (ASM2 ISPF Common Message Library)

Default: C'$PREFIX.CAIISPM'

Specifies the name of the CA-ASM2 Message library which is used for CA-ASM2 ISPF panel support and batch programs. If DD ISPMLIB is not allocated when the CA-ASM2 ISPF common message library is needed, it is allocated to the name specified by $ISRMLIB. If $PREFIX is specified as the data set name prefix, the value from the $PREFIX data item is substituted for $PREFIX, otherwise the data set name specified is allocated to the ISPMLIB DD.

3.2.3.39 $RLDGRP (Reload Group)

Default:
```
DC CL8'3380',CL8'SYS80'
DC CL8'3350',CL8'WRK3350'
DC CL8'    ',CL8' '
DC CL8'    ',CL8' '
DC CL8'    ',CL8' '
```

The above is an example only. $RLDGRP is the reload device table associated with $SELDISK option X'04'. An entry should be placed in the table (a maximum of five) for each DASD device group that the user would like to reload data to, using the nonspecific allocation capabilities of $SELDISK option X'04'. Each entry is a device group/esoteric unit name pair. When reloading a data set using option $SELDISK X'04', the device type the data set resided on at unload time is converted to the device group name, which is then matched to the table. The esoteric unit name specified in that entry is then used in allocating the data set

during reload. The exact volumes the data set are reloaded to are assigned by the Operating System.

This option should be used for **SMS** installations to allow SMS to select the actual reload volume. When the data set being reloaded is determined to be SMS controlled, the UNITNAME presented is ignored and SMS performs the actual volume selection. If the system ACS routines do not identify the data set as SMS controlled, the normal system allocation routines select a volume from the UNITNAME specified. In all cases, the CA-ASM2 IPC record is updated to reflect the actual volsers the data set is reloaded to.

Note: The $PDM data mover requires that the reload volume assigned by the system be of the same device type that the data set was unloaded from. For this reason, care should be taken in coding the $RLDGRP table entries. The esoteric unit name that is assigned to a specific generic unit type should contain only volumes of that unit type. For example, if SYSDA contains both 3380 and 3390 type units, do not specify SYSDA as the esoteric unit name for either 3380 or 3390. The esoteric unit name assigned to generic unit 3390 should contain only 3390 devices. This table is not required for any CA-ASM2 processing other than the $SELDISK X'04' reload option.

CA-ASM2 also supports allocation by UNITNAME where the user selects the UNITNAME in the $SELVOLX and $IXRUSVX exits. For a detailed description of these exits, see the *CA-ASM2 System Reference Guide*.

3.2.3.40 $DSAUNIT (Disk Staging Archive Unit)

Default: DC CL8'ARCHIVE'

Specifies the MVS esoteric unit name for allocations into the Disk Staging Area (DSA). This esoteric is used when $DSAUNIT is specified in an unload run without an esoteric unit specified or with $OPT specified.

3.2.3.41 $DSACLAS (Disk Staging Archive Class)

Default: DC CL30'ARCHIVE'

This is the SMS storage class name for allocations into the DSA. It is used when $DSACLAS is specified in an unload run without an SMS class name specified. If the $DSACLAS parameter does not specify a class name and the $DSACLAS field in $OPTIONS is also blank, the allocation is done with no class name specified and the SMS ACS routines must be setup to assign a storage class name.

3.2.3.42 $DSADAYS (Disk Staging Archive Days)

Default: DC H'60'

Specifies the default number of days each data set is to be maintained in the DSA before being eligible for purge without the DSA filling up. This parameter applies to both non-SMS data sets and SMS managed data sets.

3.2.3.43 $DSBUNIT (Disk Staging Backup Unit)

Default: DC CL8'BACKUP'

Specifies the MVS esoteric unit name for allocations into the Disk Staging Area (DSA). This esoteric is used when $DSBUNIT is specified in an unload run without an esoteric unit specified or with $OPT specified.

3.2.3.44 $DSBCLAS (Disk Staging Backup Class)

Default: DC CL30'BACKUP'

This is the SMS storage class name for allocations into the DSA. It is used when $DSBCLAS is specified in an unload run without an SMS class name specified. If the $DSBCLAS parameter does not specify a class name and the $DSBCLAS field in $OPTIONS is also blank, the allocation is done with no class name specified and the SMS ACS routines must be setup to assign a storage class name.

3.2.3.45 $DSBDAYS (Disk Staging Backup Days)

Default: DC H'10'

Specifies the default number of days each data set is to be maintained in the DSA before being eligible for purge without the DSA filling up. This parameter applies to both non-SMS data sets and SMS managed data sets.

3.2.3.46 $DPREFIX (DSA Prefix)

Default: DC C'CAI.ASM2D

Enter a DC C'XXXX' where XXXX is the CA-ASM2 data set prefix to use for data sets unloaded into a CA-ASM2 Disk Staging Area. **Do not include a trailing period**. The maximum size of the CA-ASM2 prefix is 21 characters.

3.2.3.47 $CMPTYPE (Compression Type)

Default: DC CL1'S'

Specifies the type of data compression to be applied to data being written to tape or DSA.

S Specifies simple compression which suppresses repeating characters.

F Specifies fast compression which suppresses certain repeating characters without a lot of CPU time.

T Specifies text compression which suppresses any repeating character and squeezes text data into 6-bit codes.

3.2.3.48 $THRESH (Threshold Default)

Default: PL2'95'

Specifies the default threshold that triggers automatic archival. The number specified represents the percentage at which a monitored volume or storage pool is considered to be overutilized or nearing capacity. This field is used by the RealTime Space Monitor ASM2SMON procedure when a MONITOR control statement specifies THRESHOLD=($OPT,XXX) as the trigger threshold value. Also see $DOWNTO, described next.

3.2.3.49 $DOWNTO (Down Threshold Option)

Default: PL2'85'

Indicates the default threshold that $DOWNTO($OPT) is specified on the archive command **OR** the percent value to be subtracted from 100 percent to compute $DOWNTO when $FREE($OPT) is specified.

3.2.3.50 $SASMOD (SAS Module Name)

Default: CL8'SAS'

Specifies the name that CA-ASM2 uses to invoke the SAS program.

3.3 General Protect Criteria

The Controlled Scratch component of CA-ASM2 can automatically remove data sets based on user-defined criteria when those data sets are no longer needed. Once criteria for controlled scratching have been established, CA-ASM2 determines which data sets are to be scratched and automatically scratches them at the designated time.

The Controlled Scratch component is both powerful and flexible and may be customized for your environment in a number of ways. You may choose to defer the customization of Controlled Scratch until a later time. If so, simply accept the defaults provided on the ASM2PROT macro, this builds control tables which do not allow Controlled Scratch to scratch any data sets until you return to customize them and activate scratch processing.

Controlled Scratch offers two methods of automatically scratching data sets using predetermined criteria:

1. Specifying criteria for the selection of data sets to be scratched and running the $SCRATCH program, or
2. Specifying criteria to identify data sets to be protected from scratch and executing the $PROTECT program.

The $SCRATCH program simply scratches those data sets specified by user criteria such as date of last use.

The $PROTECT program scratches all data sets **except** those protected by user-supplied criteria.

Data set protection is provided in four ways:

- CA-ASM2 built-in protection
- Protect data set
- User exits
- General protect criteria (specified at installation time)

The CA-ASM2 built-in protection, Protect data set, and user exits for Controlled Scratch are described in Chapter 4 of the *CA-ASM2 System Reference Guide*.

General protect criteria is **only** used when the $PROTECT program is run. **(It is not looked at when $SCRATCH is specified.)** General protect criteria is specified through parameters in the ASM2PROT macro during the base product installation. If general protect criteria overrides are valid, then general protect criteria can be specified for each run of the $PROTECT program. Whether or not general protect criteria overrides are valid is controlled by the OVRRIDE parameter of the ASM2PROT macro. General protect overrides may then be specified when the $PROTECT program is executed. (See example associated with OVRRIDE example.)

General protect criteria possibilities are explained next. Use the General Protect Criteria on A-7 to fill in the parameters that you desire for your system. You can use the worksheet to specify your selections during the base product installation.

Note: If you do not change the default settings for the protection criteria during installation, no controlled scratching of data sets can take place (see MAXSCR= and PCTSCR=).

PFILE=

Specifies whether data set entries in the Protect Data Set are to be protected from selection for Controlled Scratch.

YES Indicates the Protect Data Set entries are exempt from controlled scratch. **YES** is the default.

NO Indicates the Protect Data Set entries are processed for selection by controlled scratch.

HINDX=(name1,...)

Specifies one or more high-level indexes. All data sets containing specified indexes are protected.

SINDX=(name1,...)

Specifies one or more second-level indexes. Any data set containing one of the specified second-level indexes is protected.

GDG=

Specifies the level of protection given specifically for generation data group data sets.

NO Indicates that no special consideration is given to GDG data sets, which are protected through other specified criteria. **NO** is the default.

CATLG Specifies that all GDG data sets are protected if cataloged.

ALL Specifies that all generation data group data sets are protected.

NONTEMP=

Specifies the level of protection given specifically for nontemporary data sets in addition to other criteria.

NO Indicates no special protection consideration for nontemporary data sets. **NO** is the default.

CATLG Specifies that all nontemporary data sets are protected if they are cataloged.

ALL Specifies that all nontemporary data sets are protected.

AGE= nnn

Specifies that data sets nnn days old or less are to be automatically protected. The maximum value permitted is 365. If AGE=0 is specified, data sets are not selected

for scratch until the day after creation. 0 is the default. (Based on date changing at midnight.)

Example:

A data set is created on 10/11/99 and AGE=1 is specified. The data set is not selected for scratch until 10/13/99.

USE=

Specifies that data sets accessed in the last nnn days are automatically protected. The maximum value permitted is 365. 0 is the default. (Based on date changing at midnight.)

Example:

A data set was last opened on 10/10/99 and USE=2 is specified. In the Scratch Selection run on 10/12/99, the data set is not selected. pt.EXPDT=

Specifies whether data sets are to be protected by expiration date.

NO Indicates data sets are not protected by expiration date. **NO** is the default.

YES Specifies that data sets are protected if the respective expiration dates have not passed.

MAXSCR=

Specifies the maximum number of data sets which can be selected for Controlled Scratch. If this number is exceeded, no data sets are selected for Controlled Scratch. The maximum value permitted is 9999. 0 is the default which allows no scratching to take place.

PCTSCR=

Specifies the maximum percent of all data sets analyzed that can be selected for Controlled Scratch. If this value is exceeded, no data sets are selected for Controlled Scratch. The maximum value permitted is 100. 0 is the default which allows no scratching to take place.

CVOL=(vol1,...)

This parameter is used for OS CVOL support. If your user catalogs are accessible through the master catalog, you need not specify this parameter. If not, specify the catalog volumes by volume serial number, other than the system resident volumes, that CA-ASM2 must search for catalog information.

OVRRIDE=(PFILE,GDG,NONTEMP,AGE,USE,EXPDT,MAXSCR,PCTSCR)

Specifies which general protect criteria can be overridden (if any) when executing the $PROTECT command.

Example: If the AGE parameter of the ASM2PROT macro is set to 30, all data sets 30 days old or less are protected. If AGE is specified through the OVRRIDE parameter as being allowed to be overridden, when the $PROTECT command is issued for a particular run, the AGE parameter associated with the $PROTECT command may be specified as something different than 30. If, when the

$PROTECT command is again executed and the AGE parameter associated with the $PROTECT command is not specified, then the general protect criteria parameter AGE of 30 would again be the default. **Any criteria not specified by OVRRIDE cannot be overridden during execution of the $PROTECT command. The general protect criteria remains in effect.**

Use of the $PROTECT command is described in the *CA-ASM2 System Reference Guide*.

3.4 Base Installation Overview

Following is a short overview of the steps involved with the installation of the CA-ASM2 base product. Each of these steps is outlined in detail in *CA-ASM2 Getting Started*. You can install IXR and Allocation Manager immediately after installing the base product or you can install these features at a later date.

Step	Description
1	Down load sample JCL from installation tape to disk.
2	Edit the ISPF L1CHANGE macro with global variables from your worksheet.
3	Allocate the Computer Associates common target libraries.
4	Allocate the Computer Associates common SMP/E data sets.
5	Tailor the CA-ASM2 SMP/E procedure.
6	Allocate CA-ASM2 distribution libraries and libraries not controlled by SMP/E.
7	Unload data sets not controlled by SMP/E.
8	RECEIVE the selected functions.
9	APPLY the selected functions.
10	ACCEPT the selected functions.
11	Specify your selections from the $OPTIONS worksheet.
12	Update the Quiet OPEN table.
13	Install the CA-ASM2 procedures.
14	Allocate and initialize CA-ASM2 data sets.
15	Edit and install the OPEN modification with SMP/E.
16	Edit and install the CLOSE modification with SMP/E.
17	Add CA-ASM2 commands to the TSO Command Authorization.
18	Update the Program Properties table.
19	Specify your General Protect Criteria worksheet selections.

3.5 Options ($OPTIONS) Affecting IXR

The important fields in CA-ASM2 $OPTIONS that affect reload volume selection are:

$CVLEXP
$CVLFLG
$SOFTWAR (bit x'80')
$SELDISK
$HOMEVOL
$TYPEREQ
$VOLGRP
$RLDGRP

$CVLEXP AND $CVLFLG work together under JES3 to determine when and how IXR selects, from its master Candidate Volume List (CVOL), a list of candidate reload volumes for a particular data set. At reload time, IXR selects the home volume or the volume with the largest single free extent, depending on the setting of $CVLFLG.

Note: $CVLEXP and $CVLFLG affect only IXR (not base CA-ASM2) and $CVLFLG works only under JES3. Under JES2, CA-ASM2 ignores $CVLFLG.

Likewise, $SOFTWAR, $SELDISK, $HOMEVOL, $TYPEREQ, $VOLGRP, and $RLDGRP work together under JES2 to determine how IXR (as well as base CA-ASM2) chooses the target reload volume for a data set. For IXR, $CVLEXP determines how long the chosen volume remains valid.

If you want IXR to handle reloads differently from the way base CA-ASM2 handles them, you must make a copy of $OPTIONS and change the desired fields in the copy:

1. Use IEBCOPY or SPF EDIT to duplicate $OPTIONS within the CA-ASM2 CAI.PARMLIB library. Name the copy $OPTIONx where x is any alphanumeric or national character other than S.
2. Change the desired fields in $OPTIONx.
3. Assemble and link edit $OPTIONx into the CA-ASM2 load library by modifying and running job L140IH in SAMPJCL. Be sure to modify all occurrences of your old $OPTIONS name to $OPTIONx.
4. Add the statement: $OPTIONx DD DUMMY, to the job that starts IXR, IXRAJCL.

You can also add functional code to the IXR Volume Selection Exit, $IXRUSVX, or the IXR Candidate Volume Selection Exit, $IXRUCVX. These can override the fields in $OPTIONS (or $OPTIONx) in selected cases. See the chapter "Controlled Scratch" in the *CA-ASM2 System Reference Guide* for more information on User Exits.

Two other fields are important: $SMFID and $IXROPT. $SMFID determines whether IXR is to write an SMF record for each reload. $IXROPT can prevent IXR from being locked out of the SYSZTIOT ENQ in its own address space while waiting for an archive tape to be mounted.

3.6 IXR Installation Parameters

The installation parameters are used to generate JCL to install the CA-ASM2 IXR component on your operating system. These parameters are edited during the installation of IXR to fit your installation's unique environment. Multiple copies of IXR may be run in a multi-CPU shared DASD complex. See 2.8, “Multi-CPU Installation Considerations” on page 2-18 in Chapter 2, “General Planning” for further information on this topic.

Use the IXR Installation worksheet on A-8 to fill in the parameters that you desire for your system(s). A description of each of the parameters follows.

Several of the requested parameters can be obtained from the L1MODS and L1DUMPT jobs described on 3-2.

FMID401

Provide the FMID of the owner of IEFDB401.

CSECT401

Provide the CSECT name of your IEFDB401. This parameter is required only if an additional IEFDB401 exit is to be installed with IXR's IEFDB401.

LMOD401

Provide the load module name of your IEFDB401. This parameter is required only if an additional IEFDB401 exit is to be installed with IXR's IEFDB401.

FMID21SD

Provide the FMID of the component owning IEFW21SD. This parameter is only required for pre-XA users installing IXR's IEFDB401.

FMID26

Provide the FMID of the component owning IGG026DU.

CSECT26

Provide the CSECT name of the CSECT specified in your IGG026DU routine. (This is usually IGG026DU.) This parameter is only needed if you are installing an additional IGG026DU routine.

LMOD26

Provide the load module name of your IGG026DU routine. This parameter is only needed if you are installing an additional IGG026DU routine.

USERLIB26

Provide the ddname of the data set where the load module described above resides. This parameter is only needed if you are installing an additional IGG026DU routine.

FMIDX0A

Provide the FMID of the component owning IFG0EX0A.

3.7 IXR Startup Parameters

The startup parameters associated with IXR allow you to customize CA-ASM2 to operate in your unique environment. This is accomplished by specifying various parameters during the installation.

The default values shipped with the product in member IXRPARM of CAI.ASM2.R42.PARMLIB work but need to be modified for your particular needs. Each of the startup parameters is described next. Read each of the descriptions and determine whether the default condition suits your needs. If not, select your choice on the Startup Parameters Worksheet on A-9. Normally you can go with the defaults provided, but particular attention should be paid to the 522T, PTAP, TTAP, MAXDARLD and DEVTP parameters.

The parameter notation is as follows. Columns 1-71 of each record are usable. Each parameter is a keyword item, most of them taking parenthesized values or sublists. You can code multiple keyword items on a single input record. Continue a keyword whose sublist is too long for a single record by respecifying the keyword on the next record and then continuing the sublist in parentheses. See the examples illustrated on 3-58.

DSNE(lmod)

Specifies the load module name of the IXR user data set exit routine. The lmod is the one- to eight- character alphanumeric load module name. Default name is $IXRUDSX. This module must be globally accessible in LPA or MLPA.

OPTE(lmod)

Specifies the load module name of the IXR user options exit routine. The lmod is the one- to eight-character alphanumeric load module name. Default name is $IXRUROX. This module must be globally accessible in LPA or MLPA.

SELVE(lmod)

Specifies the load module name of the IXR disk volume selection exit routine. The lmod is the one- to eight-character alphanumeric load module name. Default name is $IXRUSVX. This module is required only within IXR and need not be in LPA.

Note: The startup parameters OPTBD, OPTBA, OPTBO, OPTSD, OPTSA, OPTSO, OPTTA, OPTTD, and OPTT0 are identical in syntax. The parameter keyword is followed by a parenthesized list of IXR processing option acronyms. The IXR processing options and acronyms are discussed in the *CA-ASM2 System Reference Guide*.

OPTBD(opt,opt,...)

Specifies default IXR processing options for batch job address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTBA(opt,opt,...)

Specifies AND-override IXR processing options for batch job address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTBO(opt,opt,...)

Specifies OR-override IXR processing options for batch job address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTTD(opt,opt,...)

Specifies default IXR processing options for time-sharing address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTTA(opt,opt,...)

Specifies AND-override IXR processing options for time-sharing address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTTO(opt,opt,...)

Specifies OR-override IXR processing options for time-sharing address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTSD(opt,opt,...)

Specifies default IXR processing options for started task address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTSA(opt,opt,...)

Specifies AND-override IXR processing options for started task address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

OPTSO(opt,opt,...)

Specifies OR-override IXR processing options for started task address spaces. The opt values are option Y, N, and hex priority values as discussed in the next topic. Default values are also discussed.

CONDP(char)

Specifies the MVS console area to which the results of IXR display commands are directed. The char is the one-character console area ID. Defaults to Z, which designates the rollable area of a display console.

CMDC(char)

Specifies the leading character that uniquely identifies all IXR system commands. The char is a single character that is **not** a blank or parenthesis. Default command

character is =. If this character conflicts with any existing commands, choose an alternative from the following:

```
|@#%¬-_+?!./¢
```

Be certain that all of your system console keyboards are equipped with the symbol you choose.

522T(num)

Specifies the periodic activation interval used to prevent S522 abends in address spaces waiting on completion of synchronous IXR operations. The num is the interval duration in hundredths of seconds. Default is 18000, which is three minutes. If you specify a value of zero, the S522 abend prevention feature is inoperative for all address spaces.

522I(num)

Specifies the number of periodic activations that take place between "IXR waiting" messages. The num is the number of 522T intervals (see above). This parameter is meaningful only if 522I has a nonzero value. It allows occasional message output (IXR0313I WAITING ON IXR) to indicate system status to a waiting user. Default value is one. If you specify a value of zero, no messages appear for users waiting on IXR. However, this does not disable the 522 abend prevention provided by 522T.

XSPOLL(num)

Specifies IXR's IPC polling interval for cross-system reload completion monitoring. num is the interval duration in hundredths of seconds. The polling interval determines how frequently IXR rereads the IPC record for a given archive to check the progress of a reload in progress on a different CPU. A value less than 3000 (30 seconds) causes excessive catalog accesses. Values above 60000 (10 minutes) impair IXR responsiveness in cross-system reload situations. Defaults to 18000 (3 minutes).

XSPMAX(num)

Specifies the time-out interval for cross-system reloads. The num is the interval in hundredths of seconds. The time-out interval is the length of time IXR holds a given reload request waiting for completion of a reload on a different system. If the time-out interval is reached with no change in status of the other system's reload, the waiting request is completed with error codes.

CSA(num)

Specifies the maximum aggregate amount of MVS CSA or extended CSA on XA (subpools 231 and 241) that IXR is allowed to acquire for any purpose. The num is the number of bytes. This control takes precedence over the individual CSA control block allocations set by the following parameters. Default for CSA is 65536 (64K). For CSA requirements and control, see 2-47.

ASBP(num)

Specifies the primary number of IXR address-space-related blocks to be allocated in MVS CSA or extended CSA on XA. The num is the number of blocks. This

parameter, the following two parameters (ASBS and ASBM), and the CSA parameter determine the total number of blocks that can be allocated. One address space block should be available for each MVS address space using IXR at a given instant. The total allocation of address space blocks must never exceed your system's MAXUSER value (in member IEASYSxx of SYS1.PARMLIB). Default is 15. For CSA requirements and control, see 2-47.

ASBS(num)

Specifies the secondary allocation quantity of IXR address-space-related blocks in MVS CSA or extended CSA on XA. The num is the number of blocks per secondary allocation. The maximum number of secondary allocations is controlled by the ASBM parameter and may also be controlled by the CSA parameter. Default is 5. For CSA requirements and control, see 2-47.

ASBM(num)

Specifies the maximum number of secondary allocations the address-space-related blocks are permitted to acquire. The num is the number of secondary allocations. If you specify zero, only the primary quantity (ASBP) of address space blocks are allocated. Default is 10.

ASBW(pct)

Specifies the usage percentage of IXR address-space-related blocks that generate a warning message to the system console. The pct is a percent number between 1 and 100. Default is 80.

XRBP(num)

Specifies the primary number of IXR request-related blocks to be allocated in MVS CSA or extended CSA on XA. The num is the number of blocks. This parameter, the following two parameters (XRBS and XRBM), and the CSA parameter determine the total number of blocks that can be allocated. IXR requires one request block for each currently active IXR request. Normally, this number does not exceed, and is much lower than, the number of address spaces (MAXUSER value) in your system. Default is 15. For CSA requirements and control, see 2-47.

XRBS(num)

Specifies the secondary allocation quantity of IXR request-related blocks in MVS CSA or extended CSA on XA. The num is the number of blocks per secondary allocation. The maximum number of secondary allocations is controlled by the following parameter (XRBM) and may also be controlled by the CSA parameter. Default is 5. For CSA requirements and control, see 2-47.

XRBM(num)

Specifies the maximum number of secondary allocations the request-related blocks can acquire. The num is the number of secondary allocations. If you specify zero, only the primary quantity (XRBP) of request blocks is allocated. Default is 10.

XRBW(pct)

Specifies the usage percentage of IXR request-related blocks that generate a warning message to the system console. The pct is a percent number between 1 and 100. Default is 80.

DSBP(num)

Specifies the primary number of IXR data-set-related blocks allocated in MVS CSA or extended CSA on XA. The num is the number of blocks. This parameter, the following two parameters (DSBS and DSBM), and the CSA parameter determine the total number of blocks that can be allocated. IXR requires one data set block for each data set associated with each currently active IXR request. In systems without job-level or step-level staging, this number does not exceed the number of concurrent requests (XRBP/XRBS/XRBM values). In systems with job-level or step-level staging, the data set block requirement is a function of the number of DD statements in jobs and the number of jobs entering initiation concurrently at a given moment. A value two to four times the number of request blocks is suggested as a starting point. Default is 50. For CSA requirements and control, see 2-47.

DSBS(num)

Specifies the secondary allocation quantity of IXR data-set-related blocks in MVS CSA or extended CSA on XA. The num is the number of blocks per secondary allocation. The maximum number of secondary allocations is controlled by the following parameter (DSBM) and may also be controlled by the CSA parameter. Default is 20. For CSA requirements and control, see 2-47.

DSBM(num)

Specifies the maximum number of secondary allocations the data-set-related blocks can acquire. The num is the number of secondary allocations. If you specify zero, only the primary quantity (DSBP) of data set blocks is allocated. Default is 10.

DSBW(pct)

Specifies the usage percentage of IXR request-related blocks that generate a warning message to the system console. The pct is a percent number between 1 and 100. Default is 80.

DDPRE(chars)

Specifies the prefix that uniquely identifies all IXR option DD statements or allocations. The chars is the two-character alphanumeric prefix in which the first character must be an alphabetic or national character. Default is X$.

PTAP(num)

Specifies the number of permanent tape devices dynamically allocated by IXR. The num is an integer from 0 to 31. If you want permanent tapes, the TTAP and TCUU parameters described next are an alternative method of specifying them. TTAP allows you to indicate some number of temporary tape drives to allocate dynamically as needed. TCUU allows you to indicate particular device addresses. The devices allocated for PTAP are not at any particular chosen addresses. If you

specify or default zero for PTAP and do not use TCUU, the value specified for TTAP (shown next) must be nonzero. Permanent tapes are currently supported only in MVS/SP systems at the 1.3.2 level or higher. Default is zero.

IXR uses the DEVTP startup parameter to select the MVS esoteric unit name to use for permanent tape devices.

TTAP(num)

Specifies the maximum number of temporary tape devices that can be used by IXR. The num is an integer value from 0 to 31. IXR uses the $TPUNIT table in $OPTIONS to select the MVS esoteric unit name to use for temporary tape drive allocations to correspond to the media type to which the data set was unloaded.

TCUU(dev,dev,...)

Specifies device addresses for tapes to be used by IXR as permanent tapes. The dev,dev,... is a list of 1 to 31 three-digit hexadecimal tape device addresses with no duplications. The PTAP parameter above is an alternative method for specifying permanent tapes that does not require device addresses. Permanent tapes are currently supported only in MVS/SP systems at the 1.3.2 level or higher. Default is no devices.

DEVTP(name)

Specifies the MVS esoteric unit name to use at IXR startup for permanent IXR tape device allocations. The name is the one- to eight-character MVS unit name. Default is TAPE.

PVOL(ser,ser,...)

Specifies volsers that are recognized by IXR as "fictitious." The ser,ser,... is a list of up to 32 one- to six-character alphanumeric volsers. Specify all fictitious volsers in use for CA-ASM2 archive processing in IXR's PVOL parameter. There is no default for PVOL.

Note: MIGRAT should not be specified when recataloging a data set to a fictitious volser. Some IBM software expects data sets recataloged to volsers with MIGRAT to have been archived with DFHSM, and takes appropriate action.

CVOL(ser,ser,...)

Specifies volsers to be included in the IXR master Candidate Volume List for JES3 support. The ser,ser,... is a list of up to 32 one- to six-character alphanumeric volsers of the DASD volumes at your site to which IXR may reload archived data sets under JES3. The volumes need not reside on the same type of DASD, but they all must be accessible from a single CPU. There is no default for CVOL.

Note that CVOL(*) is not valid in the IXR startup parameters.

UCSX(lmod)

Specifies the load module name of the IXR Candidate Volume Exit routine for JES3 support. The lmod can be up to eight alphanumeric characters. Default name

is $IXRUCVX. This module must be globally accessible, located in LPA or MLPA.

STOPX37(YES/NO)

Specifies whether STOPX37 is installed and can be used for cataloged PS and PO reloads. The default is NO. (STOPX37 is a product that stops B37, D37 and E37 space abends and VSAM out-of-space conditions.)

BACKUPS(NO/YES)

Specifies whether IXR is to consider backup unloads as reload candidates. The default and normal mode should be NO. YES causes IXR to select the most recent archive or backup unload. This could be useful in a disaster recovery situation.

MAXDARLD(num)

Specifies the maximum number of simultaneous reloads to be performed from the CA-ASM2 Disk Staging Area. The number is an integer from 0 to 9999. The default is 3. This functions similarly to the limitations placed on tape reloads and helps control system resource utilization and DSA device performance.

RLDTMPNM(STD/ALT)

Specifies whether the temporary data set name used by IXR during a reload has just the first high-level qualifier of the original data set name or consists of either the first three high-level qualifiers for non-VSAM data sets, or the first two high-level qualifiers for VSAM data sets. The default is STD.

For example:

A job is executed that requires USRNODE1.USRNODE2.USRNODE3.USER to be reloaded via IXR. This data set is a non-VSAM data set. With RLDTMPNM(STD), the temporary data set name that IXR generates is:

```
USRNODE1.ASM2IXR.Tnnnnnn.Dnnnnnn
```

If this same data set is a VSAM data set the temporary names that IXR generates are:

```
USRNODE1.ASM2IXR.Tnnnnnn.Dnnnnnn          (VSAM cluster name)
USRNODE1.ASM2IXR.Tnnnnnn.Dnnnnnn.DATA     (VSAM data component name)
USRNODE1.ASM2IXR.Tnnnnnn.Dnnnnnn.INDEX    (VSAM index component name)
USRNODE1.ASM2IXR.Tnnnnnn.Dnnnnnn.AIXnn    (VSAM alternate index component name)
USRNODE1.ASM2IXR.Tnnnnnn.Dnnnnnn.PATHnn   (VSAM path name)
```

With the same job and data set being reloaded using RLDTMPNM(ALT), the following IXR temporary data set name is generated for a non-VSAM data set:

```
USRNODE1.USERNODE2.USRNODE3.ASM2IXR.Dnnnnnn
```

If the same data set is a VSAM data set, the temporary names that IXR generates are:

```
USRNODE1.USRNODE2.ASM2IXR.Dnnnnnn          (VSAM cluster name)
USRNODE1.USRNODE2.ASM2IXR.Dnnnnnn.DATA     (VSAM data component name)
USRNODE1.USRNODE2.ASM2IXR.Dnnnnnn.INDEX    (VSAM index component name)
USRNODE1.USRNODE2.ASM2IXR.Dnnnnnn.AIXnn    (VSAM alternate index componet name)
USRNODE1.USRNODE2.ASM2IXR.Dnnnnnn.PATHnn   (VSAM path name)
```

3.7.1 Default and Override Option Parameters

Default option parameters OPTTD, OPTBD, and OPTSD have default settings. If you do not specify values for each parameter, IXR assumes the following default values:

```
OPTTD: IXRN,UJIN,USIY,S99Y,26AY,26JN,26VY,213Y,DFRN,
         INTY,QUIN,522Y,CNDN,ASYN,IASY,SECN,
         FRCN,VERN,JIP30,SIP30,99P30,2AP30.2VP30,13P30

OPTBD: IXRN,UJIY,USIN,S99Y,26AY,26JN,26VY,213Y,DFRN,
         INTY,QUIN,522Y,CNDN,ASYN,IASN,SECN,
         FRCN,VERN,JIP10,SIP10,99P10,2AP10,2VP10,13P10

OPTSD: IXRN,UJIN,USIY,S99Y,26AY,26JN,26VY,213Y,DFRN,
         INTN,QUIN,522Y,CNDN,ASYN,IASN,SECN,
         FRCN,VERN,JIP20,SIP20,99P20,2AP20,2VP20,13P20
```

IXR assumes that any option items omitted from the OPTxD parameters are N, meaning off or in the case of request priorities, zero. So, if you want to set an option to N, you do not need to specify it. For example, if you place in your IXR startup parameters the single statement

```
OPTTD(IASN)
```

IXR startup assumes you mean

```
OPTTD(IXRN,UJIN,USIN,S99N,26AN,26JN,26VN,213N,DFRN)
OPTTD(INTN,QUIN,522N,CNDN,ASYN,IASN,SECN)
OPTTD(FRCN,VERN,JIP00,SIP00,99P00,2AP00,2VP00,13P00)
```

To reverse the IAS option and still retain the other defaults as listed above, you must specify:

```
OPTTD(IXRN,UJIN,USIY,S99Y,26AY,26JN,26VY,213Y,DFRN)
OPTTD(INTY,QUIN,522Y,CNDN,ASYN,IASN,SECN)
OPTTD(FRCN,VERN,JIP30,SIP30,99P30,2AP30,2VP30,13P30)
```

Remember, you can omit all options to be set as N.

The AND/OR override option parameters behave similarly to the default options. If you do not code these parameters, IXR's default for AND-overrides is all on and for OR-overrides is all off. This effectively leaves the combination of defaults and user overrides undisturbed. If you want to disable a particular option, specify its AND setting as N. To activate an option, specify both its AND and OR settings as Y.

If you specify a value for either an AND or an OR group, you **must** specify the entire set of options. For example, to disable only the IAS option for TSO users, specify:

```
OPTTA(IXRY,UJIY,USIY,S99Y,26AY,26JY,26VY,213Y,DFRY)
OPTTA(INTY,QUIY,522Y,CNDY,ASYY,IASN,SECY)
OPTTA(FRCY,VERY,JIPFF,SIPFF,99PFF,2APFF,2VPFF,13PFF)
```

3.8 IXR Installation Overview

This section is designed to give you an overview of the major parts to installing IXR:

- Load Modules
- MVS Exits/Modules
- IXR Subsystem

3.8.1 Load Modules

IXR has two groups of load modules: global and private. Global modules are used by IXR intercepts, so they must be accessible from any address space. Private modules are used only within the subsystem itself.

3.8.2 MVS Exits/Modules

IXR provides intercepts at five system exit points. The Catalog Management (IGG026DU) and Dynamic Allocation (IEFDB401) exit points must be installed as USERMODs.

USERMODs are provided to install:

- IXR job initiation exit (IEFUJI)
- Step initiation exit (IEFUSI)
- Dynamic Allocation (IEFDB401)
- Catalog Management (IGG026DU)
- OPEN-S213 (IFG0EX0A)

Your site may have already installed various exit routines it designed or got from other products. In the install procedure for IEFDB401 and IGG026DU, IXR provides for the existence of more than one exit routine at the same logical point. With IEFDB401 IXR uses a driver version of the exit that, in turn, calls each of several variations of the exit with the usual parameters and conditions. You can determine the order and conditions in which the driver invokes each version of the exit. For the Catalog Management Exit, IGG026DU, a link-edit technique handles multiple routines.

Because of the potentially serious impact of a problem in one of these exits, Computer Associates recommends that you initially set up these exits using the MLPA facility of MVS. Then, if problems occur, an IPL without the MLPA specified provides a quick circumvention until you can diagnose and correct the problem. After a suitable break-in time, you can safely move the exit modules to SYS1.LPALIB.

Note: Having two versions of a module, one in active MLPA and the other in regular LPA, causes an increase in MVS common (LPA) storage requirements.

IEFUJI and IEFUSI are installed with the CAIRIM utility. The CAISMFI Dynamic SMF Interceptor program (a component of CAIRIM) becomes the driver. CAISMFI invokes the SMF exits in the order in which the products are specified in CAIRIM initialization control statements (member CARIMPRM of data set CAI.PPOPTION).

Any user-written exits or those for other products are invoked after those defined to CAIRIM.

Note: The IXR job- and step-level staging should never both be turned on. This would double the overhead as all reloads would occur in job-level staging but step-level staging analysis would still occur.

3.8.3 IXR Subsystem

To set up the IXR subsystem, you must perform the following steps:

1. Determine the IXR startup parameters and set them up in a suitable library. Default startup parameters are distributed in member IXRPARM of data set CAI.ASM2.R42.PARMLIB.
2. Add the name of the primary IXR subsystem module, $IXRMAIN, to the MVS Program Properties Table (PPT) to make IXR nonswappable. Assign it a unique storage protect key, and skip the password request to the operator for any password-protected data sets that IXR reloads. The PPT updates for IXR are included with those for the base product. When you perform the PPT update for the base product, the IXR requirements are specified at that time.

3.8.4 IXR Reinstallation Considerations

If your installation already has IXR installed, you are required to reinstall the intercepts to take advantage of the CAIRIM installation technique. In addition, the new IXR procedure, IXRASUBS must be used also; new DD statements are required in the procedure for the Integrated Product Catalog (IPC) and Journal and the ISPF message data set CAI.CAIISPM. Review the other installation steps to ensure that your current system parameters are carried over to the new version (IXR installation parameters are now located in the new CAI.ASM2.R42.PARMLIB data set. Ensure they are updated to reflect your installation requirements).

Caution

It is no longer necessary to have an entry for $IXRMRCL in IEAVTRML. Please remove this entry to avoid an S806 abend at IPL.

3.8.5 IXR Installation Steps

Following is a short overview of the steps involved with the installation of IXR. Each of these steps is described in detail in *CA-ASM2 Getting Started*.

Step	Description
1	Install the IEFDB401 Intercept.
2	Install the IGG026DU Intercept.
3	Install the IFG0EX0A Intercept.
4	Specify your selections from the IXR Startup Parameters worksheet (Optional).

3.9 Allocation Manager Installation Parameters

The default installation parameters are contained in PARMLIB and are used to generate JCL to install Allocation Manager. These parameters are edited during the installation of this component to fit your installation's unique environment.

Use the Allocation Manager Installation worksheet on A-10 to fill in the parameters that you desire for your system(s). A description of each of the parameters follows. You can use the worksheet to specify your selections during the installation.

AMSVC

Provide the Type 3 or 4 SVC number that you have selected for Allocation Manager.

TSOWORK

Specify the unit name to be used to override the UNIT parameter for allocation of nonuser ID work data sets under TSO. (The data set's first node is not equal to the user ID.) This parameter is used only in the TSO DAIR component of Allocation Manager.

TSOTEMP

Specify the unit name to be used to override the UNIT parameter for allocation of temporary data sets under TSO. This parameter is used only in the TSO DAIR component of Allocation Manager.

TSOUSER

Specify the unit name to be used to override the UNIT parameter for allocation of data sets qualified by the TSO user ID. This parameter is used only in the TSO DAIR component of Allocation Manager.

FMID4DO

Provide the FMID of the component owning IEFDB4D0. This parameter is required only if installing the TSO DAIR component.

3.10 Allocation Manager Installation Overview

Following is a short overview of the steps involved with the installation of Allocation Manager. Each of these steps is described in detail in *CA-ASM2 Getting Started*.

Step	Description
1	Apply the Allocation Manager SVC to selected modules and tables.
2	Define the units to be used for TSO DAIR processing.
3	RECEIVE and APPLY the Group Name Modification USERMOD.
4	Build an Allocation Manager table.

Chapter 4. CA-ASM2 Exit Descriptions

CA-ASM2 provides many points in its processing for exits. These exits, either user-written or provided by Computer Associates, allow you to customize CA-ASM2 to your requirements. With each new version of CA-ASM2, you are notified of any modifications to user exit parameter lists or processing that would necessitate modification or reassembly of the user exits. However, you should validate all currently utilized user exits during the testing phase of a new CA-ASM2 version installation.

The CA-ASM2 processing exits described in this chapter are listed on the following page.

Exit Name	Description
$ARCHEXT	$DASDMNT pre-UNLOAD processing
$AUTHEXT	Authority exit
$AUTHXTU	Authorization exit
$BTTMS	Bad tape exit
$CNVAGE	Convert DSCB aging information
$CTLSCDE	Scratch determination user exit
$CTLSCDM	Demand scratch user exit
$CTLSCVE	Scratch verification user exit
$DDREXIT	$DEFRAG reload DSCB exit
$DDUEXIT	$DEFRAG unload DSCB exit
$DEREXIT	$DEFRAG end-of-reload exit
$DEUEXIT	$DEFRAG end-of-unload exit
$DSREXIT	$DEFRAG start-of-reload exit
$DSUEXIT	$DEFRAG start-of-unload exit
$ENDCMDX	End of Command exit
$ENDULOD	End-of-UNLOAD $DASDMNT postprocessing
$FTEXIT	Free tape processing
$IXRUCVX	IXR Candidate Volume Selection exit
$IXRUDSX	IXR Data Set Selection exit
$IXRUROX	IXR Reload Options exit
$IXRUSVX	IXR Volume Selection
$MIGDEVT	Migrate Device Table
$MIGEXIT	$MIGRATE processing
$NTEXIT	New tape preprocessing
$PDMUEXT	Physical Data Mover unload
$RELDEXT	RELOAD postprocessing
$RXQEXIT	Queued Reload exit
$SELVOLX	RELOAD volume selection
$TSINQ	Tape System Inquiry
$UVEXIT	Update retrieval
$VLISTXT	VLIST utility exit
$VSAMRXT	VSAM rename exit
AMCUSREX	Allocation Manager user exit

Note: User exits are listed alphabetically in this chapter according to the above list. CA-RSVP user exits, (those beginning with $RS) are described in the *CA-RSVP User Guide*.

4.1 $ARCHEXT (Archival Exit)

$ARCHEXT is the ultimate decision maker in the storage maintenance process. This module is called by the $DASDMNT program before it acts upon a data set. The $ARCHEXT control statement makes it a runtime decision whether to invoke this exit at all. You can code multiple $ARCHEXTs and simply specify in the input stream the particular exit module to be used in a run.

It may be desirable to structure a user exit so that certain data sets, even though they may satisfy the normal eligibility criteria, are exempted from archival. It may also be desirable to treat another type of data set as a special class that is only scratched but never archived. You may want to force $DASDMNT to write comments to an exception log when it encounters a particular type of data set, but not to archive or disturb its location on disk.

You can satisfy all these requirements by building a user exit to $DASDMNT. $DASDMNT passes to the exit the names of all data sets. The exit indicates what action to take for each data set by a return code in register 15. Possible values are:

-2	use $PDM as the data mover for archival or backup	
0	no action	
1	list data set name and comment on exception log	
2	archive, scratch, and uncatalog - or backup	
3	scratch	(non-VSAM)
4	scratch and uncatalog	(non-VSAM)
5	release unused space in data set	(non-VSAM)

Return code 2 indicates normal CA-ASM2 processing. The results of all activity based upon this return code are reported in the main log. All other activity is reported in the exception log.

The following examples illustrate other uses of the possible return codes:

(3) Scratch, or (4) Scratch and Uncatalog: Several disk packs on the system might be dedicated to project development activity and consequently plagued by large quantities of temporary test data sets abandoned at the completion of development work. If these data sets are recognizable by a qualifier, or if they all reside on one pack, they can easily be scratched through the use of this exit. Thus, your installation can enjoy an automatic and ongoing pack maintenance facility, without relying upon programming personnel to scratch and uncatalog data sets when they are no longer in use.

(0) No Action: Certain data sets might require exemption from any of the $DASDMNT functions, but reside on aged packs that are eligible for archival. By recognizing these and passing a return code of 0 to $DASDMNT, this type of data set can be stored on the same packs with others more temporary in nature.

An example: A typical development disk pack contains a variety of 112 data sets as follows:

Sixteen data sets used for quick development/maintenance projects; qualified by TEMP, inactive for 15 days. (return code 4, scratch and uncatalog)

Eighty-four data sets used for large-scale development project; qualified by WORK, inactive for 15 days. (return code 2, archive)

Eight data sets used by Data Management personnel for control files; qualified by DBD, inactive for 15 days. (return code 0, no action)

Four data sets unidentified; no recognizable qualifier, inactive for 15 days. (return code 1, list comment in exception log)

In the above example, every data set was found to be inactive for 15 days. Your installation can specify any quantity of days of inactivity it desires or establish return codes based not upon the inactivity criteria but rather upon some other basis.

The flexibility of this exit provides your installation with an ability to exercise control over every disk-resident data set on the system.

See CAI.CAIMAC($EXTPRMD) for the DSECT definition of parameters passed to the $ARCHEXT module.

4.2 $AUTHEXT (Authority Exit)

$AUTHEXT, a front-end to all CA-ASM2 commands, validates a user's authority to issue a specific command. The function of $AUTHEXT is to provide your installation with an easily programmed capability to control the use of specific CA-ASM2 commands.

All CA-ASM2 commands link to $AUTHEXT before performing any meaningful processing. $AUTHEXT has two options:

1. Exit with a return code of zero in register 15, which effectively grants authority to use the CA-ASM2 command.
2. Exit with any other value in register 15, which effectively disallows the use of the specific CA-ASM2 command. The command terminates immediately without performing its intended function.

The $AUTHEXT distributed with CA-ASM2 supports CA-ACF2, CA-Top Secret and IBM's RACF. $AUTHEXT functions as follows:

1. If bit 1 (X'40') of $SOFTWAR in $OPTIONS is on, LINK to module $AUTHXT3 to perform CA-ACF2 checking.
2. If step 1 returns nonzero, exit, denying access.
3. If bit 5 (X'04') of $SOFTWAR in $OPTIONS is on, LINK to module $AUTHXT4 to perform CA-Top Secret or RACF checking.
4. If the previous step returns nonzero, exit, denying access.
5. If module $AUTHXTU exists, LINK to it to perform user-written validity checking.
6. If module $AUTHXTU exists and if the previous step returns nonzero, exit, denying access. In other words, if module $AUTHXTU does not exist or if the call to it returns zero, continue.
7. Exit, returning to CA-ASM2.

Computer Associates reserves the names $AUTHXT1 through $AUTHXT9 and $AUTHXTA through $AUTHXTT for use in present and future $AUTHEXT exits. Your exits can have any other names.

Input to $AUTHEXT is a 16-byte read-only area that is pointed to by register 1. The parameters passed to $AUTHXTU are identical to those passed to $AUTHEXT. There are four fields in the input area.

```
Register 1

 Offset  0: A(CMDNAME)  ----------> CMDNAME  DS  CL8'cmdname'
 Offset  4: A(CPPLADDR) ----------> CPPLADDR DS  A(0) --------> CPPL
 Offset  8: A(DSNLIST)  ----------> A(1st DS entry) ----> DS entry 1
                        ----------> A(2nd DS entry) ----> DS entry 2
                                      .                          .
                                      .                          .
                                      .                          .
                        ----------> X'80',AL3(nth DS entry) ----> DS entry n
 Offset 12: A(USERDBLW) ----------> USERDBLW DS D(0)
```

Notes:

1. The cmdname pointed to by the address at offset 0 can be the name of any CA-ASM2 command, such as $AR.
2. In the address list pointed to by offset 8, the final address is preceded by X'80',AL3 to tell $AUTHEXT that this is the final address in the list.
3. The field named USERDBLW, pointed to by offset 12, can hold any user-selected doubleword.

This map shows what each data set entry looks like:

DSENTRYD	DSECT		
$XTFLAGA	DS	X'00'	Data set name flags
$XTFIXA	EQU	X'80'	An index, not a data set name
$XTFVOLA	EQU	X'08'	Volume is present
$XTFNDSA	EQU	X'04'	New data set name is present
$XTFNVLA	EQU	X'02'	New volume is present
$XTDSNA	DS	CL44	Data set name
$XTVOLA	DS	CL6	Volser
$XTNDSNA	DS	CL44	New data set name
$XTNVOLA	DS	CL6	New volser

The following are readily accessible from the CPPL (Command Processor Parameter List): the issuing user ID or job name, the command buffer, the switches reflecting whether the user has operator/account privilege, and so forth.

CA-ASM2 activities need not be restricted. However, your installation may want to prevent certain activities, for example, a $DA command that operates on data sets not belonging to the issuer of the command. The $AUTHEXT (actually $AUTHEXT, $AUTHXT3, and $AUTHXT4) supplied by Computer Associates provides the CA-Top Secret, CA-ACF2 or RACF user with a working default environment. Source is provided for all of these so that you can further tailor these exits to your needs. The $AUTHXTU provided in CAI.CAISRC performs no useful function; it is intended as a skeleton for a user-coded routine.

4.3 $AUTHXTU (Authorization Exit)

This exit is not assembled by Computer Associates. If the user requires processing which is not handled by the CA-ACF2, RACF, or the queued reload exits, it should be handled here. This sample exit should not be assembled unless user code is added.

Parameter conventions are as follows:

- Register 1 points to a list of fullwords. The parameter list passed is described by the $XTPARMS macro and DSECTs generated by this exit. This parameter list is variable in length and the high-order bit of the last parameter is ON to indicate end of list.
- A return code of zero in register 15 indicates to allow the command.
- A return code of nonzero in register 15 indicates to disallow the command.

4.4 $BTTMS (Bad Tape Exit)

The $BTTMS exit interfaces with CA-1 after CA-ASM2 has completed an unload and a non-CA-ASM2 tape has been written to. The exit allows users who wish to have CA-ASM2 remove multivolume chains from CA-ASM2 tapes in CA-1. Since the user's exact intentions are unknown, source is provided with the assumption that user modification is highly probable. The basic function of the exit is to correct any unneeded multivolume chains in the TMC so no additional problems can occur.

The input to this exit is the address of the 6-byte volume to be freed and the address of the Format-1 DSCB of the data set being unloaded at the time (optional).

The exit reads the TMC and corrects the multivolume pointers (NEXTVOL, PREVOL, 1STVOL). It also expires all but the CA-ASM2 tape putting in the correct dsname if possible.

Return codes from this module are:

R15=0	All chains corrected
R15=4	Error correcting chains

4.5 $CNVAGE (Convert Aging Exit)

$CNVAGE is the DSCB reformatting exit. The open modification saves aging information and you can modify it as desired. This exit takes the DSCB as it is and changes it to how the CA-ASM2 programs expect their input. Modification of this exit is not required unless a user-programmed OPEN modification is installed or you are converting to IBM's Systems Managed Storage.

The $CNVAGE routine is an exit invoked by modules $ARCHIVE, $VR, and $TIMES to convert the DSCB information into the CA-ASM2 standard internal format. This allows you to have your own data where you want. The $CNVAGE routine then converts the data for CA-ASM2 programs. You can also use this method if you want to retain your current DSCB format and modifications.

CAI.CAISRC($CNVAGE) contains the exit as distributed.

4.6 $DDREXIT (Defrag DSCB Reload Exit)

This exit is called just before the DSCBs are written to the disk. You can then make changes in the DSCB if necessary. The parameters are:

parm 1 Points to the $DEFRAG common area (DFGCOM member of CAI.CAIMAC) which contains information about $DEFRAG, such as bit representations of $DEFRAG parameters

parm 2 Points to the Format-1 DSCB

parm 3 Points to the volser of target volume (6 bytes)

parm 4 Contains entry code. 0 indicates a full volume restore, 1 indicates a selective volume restore

4.7 $DDUEXIT (Defrag DSCB Unload Exit)

This exit is called just before the DSCBs are written to the tape. You can then make changes in the DSCB if necessary. The Format-1 DSCB contains the program information. The parameters are:

parm 1 Points to the $DEFRAG common area (DFGCOM member of CAI.CAIMAC, which contains information about $DEFRAG, such as bit representations of $DEFRAG parameters

parm 2 Points to the DSCBs for the data set just unloaded (F1, F1F3...F3, F1F2, or F1F2F3...F3)

parm 3 Points to the volser of original volume (6 bytes)

parm 4 Contains the entry code:

- **0** VTOC analysis; original VTOC passed to the user.
- **1** In compressed mode; the compressed DSCB is being passed.
- **2** Ddata set dump; DSCBs being written to tape at this time are copies of the original.

Note: Zero and one are the same if data sets are unmovable; 0 and 2 contain the same information.

parm 5 Location of Format-1 DSCB on disk

parm 6 Reserved

parm 7 Points to the buffer header (M2$BUFH) for unloaded data set

4.8 $DEREXIT (Defrag End-of-Reload Exit)

This exit is entered after a data set is successfully restored, either during full- or partial-volume restores (except for partial data set restores). You can use this exit to do any processing you want (for example, sending a message to the owner of the data set).

$DEREXIT is passed a standard OS-format parameter list (that is, register 1 points to a list of addresses of the parameters). There are three parameters:

parm 1 Points to the $DEFRAG common area (DFGCOM member of CAI.CAIMAC) that contains information about $DEFRAG, such as bit representations of $DEFRAG parameters

parm 2 Feserved

parm 3 Points to the DSCBs for the data set just restored (F1, F1F3...F3, F1F2, or F1F2F3...F3)

In parameter 3, there may be more than one Format-3 DSCB. The Format-1 DSCB contains information for three extents and the count of extents. Each Format-3 DSCB contains information for 13 extents.

To enable this exit, link edit the module into CAI.CAILIB or your linklist library. It should have the name $DEREXIT. When it is loaded, it must be loaded from an authorized library.

4.9 $DEUEXIT (Defrag End-of-Unload Exit)

This exit is entered after a data set is successfully unloaded, either during full- or partial-volume unloads (except for partial data set unloads). You can use this exit to do any processing you want (for example, sending a message to the owner of the data set).

$DEUEXIT is passed a standard OS-format parameter list (that is, register 1 points to a list of addresses of the parameters). There are three parameters:

parm 1 Points to the $DEFRAG common area (DFGCOM member of CAI.CAIMAC) that contains information about $DEFRAG, such as bit representations of $DEFRAG parameters

parm 2 Reserved

parm 3 Points to the DSCBs for the data set just unloaded (F1, F1F3...F3, F1F2, or F1F2F3...F3)

In parameter 3, there may be more than one Format-3 DSCB. The Format-1 DSCB contains information for three extents and the count of extents. Each Format-3 DSCB contains information for 13 extents.

To enable this exit, link edit the module into CAI.CAILIB or your linklist library. It should have the name $DEUEXIT. When it is loaded, it must be loaded from an authorized library.

4.10 $DSREXIT (Defrag Start-of-Reload Exit)

This exit allows you to decide whether or not a data set is to be restored during a selective restore operation. This responsibility is shared between this exit and $AUTHEXT, with only those data sets already allowed by $AUTHEXT being passed to $DSREXIT.

$DSREXIT is passed a standard OS-format parameter list (that is, register 1 points to a list of addresses of parameters). There are three parameters:

parm 1 Points to the $DEFRAG common area (DFGCOM member of CAI.CAIMAC) that contains information about $DEFRAG, such as bit representations of $DEFRAG parameters

parm 2 Points to the data set name (44 bytes)

parm 3 Points to the volser of target volume (6 bytes)

$DSREXIT should return zero in register 15 if the data set should be restored, and four if the data set is not to be restored.

This exit is not entered for full-volume restore operations.

To enable this exit, simply link edit the module into CAI.CAILIB or your link list library. It should have the name $DSREXIT. When it is loaded, it must be loaded from an authorized library.

4.11 $DSUEXIT (Defrag Start-of-Unload Exit)

This exit allows you to decide whether or not a data set is to be unloaded during a selective unload operation. This responsibility is shared between this exit and $AUTHEXT, with only those data sets already allowed by $AUTHEXT being passed to $DSUEXIT.

$DSUEXIT is passed a standard OS-format parameter list (that is, register 1 points to a list of addresses of parameters). There are three parameters:

parm 1 Points to the $DEFRAG common area (DFGCOM member of CAI.CAIMAC) that contains information about $DEFRAG, such as bit representations of $DEFRAG parameters

parm 2 Points to the data set name (44 bytes)

parm 3 Points to the volser of original volume (6 bytes)

$DSUEXIT should return zero in register 15 if the data set should be unloaded and four if the data set is not to be unloaded.

This exit is not entered for full-volume unload operations.

To enable this exit, simply link edit the module into CAI.CAILIB or your link list library. It should have the name $DSUEXIT. When it is loaded, it must be loaded from an authorized library.

4.12 $ENDCMDX (End-of-Command Exit)

The $ENDCMDX exit is for use in a CA-Top Secret or RACF environment. It is called to turn off special authority that had been required to process the following commands:

$DA
$DB
$RB
$RC
$RL
$UA
$UB

4.13 $ENDULOD (End-of-Unload Processing)

$ENDULOD receives control after every successful or unsuccessful move of data from disk to tape. This exit is sometimes used to perform some immediate actions upon backup or archival, such as informing you or your job stream of the results of an archival or backup action.

$ENDULOD receives control by a CALL (it must be reusable) from $DASDMNT. Control is passed in three situations:

1. After every successful unload to tape and before CA-ASM2 performs any uncatalog or scratch functions.
2. After every unsuccessful unload to tape.
3. At the end of the job step (that is, after all UNLOAD processing for the run is complete).

Computer Associates supplies an $ENDULOD load module on the installation tape. Input parameters to this exit are described in CAI.CAIMAC($EULXD).

To enable the $ENDULOD exit interface, turn on bit X'10' in the $ASM2FLG entry in $OPTIONS.

This exit may influence actions performed by mainline CA-ASM2 processing only when entered under condition (1) above. Upon entry to this exit, the three bytes at EXXRC are set to C'XXX'.

For backup runs, these bytes are not inspected by CA-ASM2.

For archive runs, however, the exit may perform its own scratch and uncatalog operations and use these bytes at EXXRC to report the results to CA-ASM2. If these bytes at EXXRC are all equal to C'X' when control is returned to CA-ASM2, it performs the required scratch and uncatalog operations. If any byte is not equal to C'X', CA-ASM2 assumes the exit performed these functions. Users may not perform their own scratch and uncatalog processing for VSAM cluster unloads, as it may have been performed during the IDCAMS EXPORT.

Under condition (2) above, the exit may not influence CA-ASM2 actions. Unsuccessful data moves are signaled by a data mover return code (EXDMRC) in the Main Log higher than C'004'.

Under condition (3) above, the exit may logically close any data sets opened by $ENDULOD or $ARCHEXT processing. This null entry is indicated by the fact that register 15 is equal to register 1 upon entry to $ENDULOD. Other parameters passed to $ENDULOD for condition (3) are meaningless.

4.14 $FTEXIT (Free Tape Exit)

$FTEXIT enables your installation to receive control immediately before CA-ASM2 frees a tape. You can cause CA-ASM2 to not free the tape. However, the more likely function is to perform some user processing such as interfacing with a tape management system.

This exit is required, and a usable version of it is supplied. By setting variables in the $XTTMS member of CAI.CAIMAC and reassembling $FTEXIT, you can create interfaces to various tape management program products.

Input Parameters:

Register 1 points to the address of the volume serial number being freed.

Return Codes (register 15):

0 - If tape pool in use, mark the tape reusable.
4 - Do not update the tape as reusable in the CA-ASM2 tape pool.

4.15 IXR User Exits

IXR has four points where your installation can programmatically alter operating logic by installing exit routines. Two of these points are executed in the address space of the intercepted user or job, and are concerned with the status of requests before they are sent to IXR for processing. The third and fourth are executed in the IXR subsystem address space and are involved in selecting a DASD volume on which to reload archived data.

Because the user-address-space exits are executed in multiple MVS memories, you must either install them in SYS1.LPALIB or make them available in the Link Pack Area by use of the MLPA facility. The exits used in the subsystem need not be in LPA. You can place them in the IXR STEPLIB, if any, or in one of your LNKLSTxx libraries.

If your installation decides to put its own exit routines in IXR, Computer Associates recommends installing them with SMP. There is no specific IXR procedure for installing installation exits with SMP. You can use SMP regardless of the way the rest of IXR was installed. Either way, your installation is responsible for the results of coding, installing, and using its own exit routines.

Whichever way they are installed, all four exits must be reentrant because IXR loads and uses only a single copy of each. All four also use certain common parameters to identify the user (or job) and data set for which the exit was invoked.

The user or job that originated the IXR request for which the exit is being invoked is described by the $UIB data area. A mapping macro for the $UIB (invoked with the macro name #UIB) is supplied in CAI.CAIMAC. For assembly purposes, you may copy the macro definition to an existing library. Or, you can concatenate (block sizes permitting) it to the SYSLIB DD during assembly of an exit. The content of the #UIB is shown following.

Field Name	Structure	Description (data source)
$UIB	DSECT	Name of mapping DSECT
$UIBID	C'$UIB'	Control block identifier
$UIJOBNM	CL8	Job name (IBM 'SCT')
$UISTEPN	CL8	Step name (IBM 'SCT')
$UIPSTPN	CL8	Procedure step name (IBM 'SCT')
$UIASID	FL2	Address space ID (IBM 'ASCB')
$UISTEP#	FL2	Current step number (IBM 'JCT')
$UITYPE	CL1 --	Address space type (IBM 'SSIB') --
$UITJOB	C'J'	-- batch job
$UITTSU	C'T'	-- time-sharing user
$UITSTC	C'S'	-- started task or mount
$UITRESC	C'R'	-- rescheduled (set for batchjobs that were requeued and canceled by the IXR reschedule feature)
$UICLASS	CL1	Job class (IBM 'SSIB')
$UIBLEN	X'22'	Length of $UIB

For the two exits that deal with specific data sets (the Data Set Selection and Volume Selection Exits), the data set being considered is identified by a control block called a $UDB. The $UDB is mapped by macro #UDB in CAI.CAIMAC. Considerations for using this macro in an assembly are the same as with the #UIB macro. The content of the #UDB is shown following.

Field Name	Structure	Description (data source)
$UDB	DSECT	Name of mapping DSECT
$UDBID	C'$UDB'	Control block identifier
$UDINID	XL1 --	IXR intercept identifier --
$UDBIUJI	X'01'	-- Job staging (IEFUJI)
$UDBIUSI	X'02'	-- Step staging (IEFUSI)
$UDBIS99	X'03'	-- Dynamic allocation (IEFDB401)
$UDBI26A	X'04'	-- Allocation LOCATE (IGG026DU)
$UDBI26V	X'05'	-- VSAM OPEN (IGG026DU)
$UDBI213	X'06'	-- Non-VSAM OPEN (IFG0EX0A)
$UDBIJCK	X'07'	-- CA/JCLCHECK
$UDBFLG0	BL1 --	Request flags --
$UDBFCND	X'80'	-- Request eligible for conditional (CND) treatment
$UDBFDEF	X'40'	-- Request eligible for deferral
$UDBFRSC	X'20'	-- This is a job-level staging request; job can be rescheduled.
$UDBDSN	CL44	Data set name coded by the user
$UDBELNM	CL8	Element name (GDS suffix, member name, ISAM extent type) or blanks
$UDBDDNM	CL8	DDname if known or blanks
$UDBGDGB	CL4	Base generation number for GDS (see $UDBFLG2)
$UDBVOL	CL6	volser coded by the user, or blanks
$UDBDEVT	XL4	Device type of $UDBVOL in UCB format
$UDBFLG1	BL1 --	Data set reference flags --
$UDBFKV	X'80'	-- Known to be VSAM object
$UDBFKN	X'40'	-- Known to be non-VSAM object
$UDBFCAT	X'20'	-- User reference is through a catalog
$UDBFSVL	X'10'	-- IXR allowed to select reload volume
$UDBFHVL	X'08'	-- On=force reload to home volume off=force reload to $UDBVOL (meaningful only if $UDBFSVL off)
$UDBFFVS	X'04'	-- Object cataloged to fictitious volser
$UDBFNAC	X'02'	-- LOCATE for data set name known to get "not found" return code
$UDBFNAV	X'01'	-- OBTAIN for DSCB known to get "not found" return code
$UDBFLG2	BL1 --	More data set reference flags --
$UDBFGDS	X'80'	-- GDS request: $UDBDSN is base name, $UDBELNM is generation index

Field Name	Structure	Description (data source)
$UDBFFBD	X'40'	-- Intercept requires reload volume and devtype feedback
$UDBFMOD	X'20'	-- DISP=MOD reference, data set may or may not exist
$UDBFBGN	X'10'	-- Base generation number for GDS was passed in $UDBGDGB field
$UDBFDFR	X'08'	-- Reload is deferrable
$UDBLEN	X'07'	-- Reserved

4.15.1 $IXRUCVX (Candidate Volume Selection)

$IXRUCVX (Candidate Volume Selection Exit) is invoked before IXR returns a list of candidate reload volumes to JES3 CI (Converter-Interpreter). This exit can change the list of candidate reload volumes selected by IXR for an archived data set. The name of the exit is set by the IXR startup parameter UCSX. The default name is $IXRUCVX.

$IXRUCVX receives control in Task Mode, in Supervisor state, and in the protect key used for the IXR Subsystem. All parameters passed to the exit are in subsystem-key storage, so they are accessible. Register contents on entry to the exit are shown following.

Register	Contents
R0	Indeterminate
R1	Address of 28-byte parameter list: Offset / Length / Description 0 / 4 / Address of $UIB control block (describes job that requested data set) 4 / 4 / Address if 512-byte work area that exit can use 8 / 4 / Address of $UDB control block (describes job that requested data set) 12 / 4 / Address where final list of candidate reload volumes for data set are written 16 / 4 / Address of copy of archive catalog record for data set 20 / 4 / Address of copy of an archive catalog extension record, or zero if none 24 / 4 / Address of master Candidate Volume List (CVOL)
R2-R12	Indeterminate
R13	Address of a conventional 72-byte save area
R14	Return address
R15	Address of exit's entry point

The exit must save and restore any of registers 2-14 that are used. For this purpose, the exit can use the save area addressed by R13. The exit need not save and restore registers 0, 1, and 15. Unless it needs more than 512 bytes of work storage, the exit can use the work area addressed at offset 4 in register 1, so the exit need not call MVS GETMAIN and FREEMAIN macros. The work area might not contain all zeros when the exit is invoked, so the exit should zero the work area before using it. Code the exit to take no action when passed a request with the job name JES3. This means that JES3 CI, not IXR, currently controls the search for data sets.

The exit can do one of three things:

1. Nothing. IXR returns the list of candidate volumes unchanged to JES3 CI.
2. Change or delete any entry in the list. If the exit changes a volser, it must also change the corresponding device type. If the exit deletes a volser, it must decrement the volume count at the front of the list.

   The exit can add a new volser if the list contains fewer than five volsers. This occurs only if the master Candidate Volume List contains fewer than five. In such a case, the exit must supply the device type of the new volume and increment the volume count.
3. Delete all entries in the list. IXR returns a data set not found to JES3 CI, and CI issues an EXPRESS CANCEL for the job that requested the data set.

4.15.2 $IXRUDSX (Data Set Selection)

$IXRUDSX (Data Set Selection Exit) is invoked for each data set associated with each IXR request. The exit is invoked only if the request processing options indicate that IXR is to process the request. The exit can only indicate whether the particular data set whose description was passed by a $UDB is to be processed by IXR.

This exit receives control in task mode, supervisor state, and the protect key of the IXR subsystem. All passed parameters are in subsystem-key storage, and thus are accessible. Register contents on entry to the exit are shown following.

Register	Contents
R0	Indeterminate
R1	Address of 12-byte parameter list: Offset / Length / Description 0 / 4 / Address of $UIB control block (describes job that requested data set) 4 / 4 / Address if 512-byte work area that exit can use 8 / 4 / Address of $UDB control block (describes data set that was requested)

Register	Contents
R2-R12	Indeterminate
R13	Address of a conventional 72-byte save area
R14	Return address
R15	Address of exit's entry point

The exit must save and restore any of registers 2-14 that are used. The save area addressed by R13 on entry can be used for this purpose. Registers 0 and 1 need not be saved and restored. Unless the exit requires more than 512 bytes of work storage, it can use the work area passed as the second parameter instead of issuing MVS GETMAIN and FREEMAIN macros. The work area is not guaranteed to be all zeros when the exit is invoked, so any fields used as counters, flags, and so forth, should be zeroed by the exit before use. Code the exit to take no action when passed a request with the job name JES3. This means that JES3 Converter-Interpreter, not IXR, currently controls the search for data sets.

Before returning, the exit must set a return code in register 15 as follows:

X'00000000' - Process the data set.
X'00000004' - Bypass processing for this data set.

Data sets that are bypassed by the exit are never seen by the IXR subsystem and do not participate in the request. If all data sets in a request are bypassed, the request is not submitted to IXR at all.

4.15.2.1 IXR Process Table

The $IXRUDSX module determines whether a given data set should be processed by IXR. This decision is made based on the Process table contained in the first portion of the $IXRUDSX module. $IXRUDSX scans this table for a match. For each entry in the table, $IXRUDSX determines whether or not the entry applies to the current requesting address space. If the entry applies, $IXRUDSX checks to see that other criteria relating to the job name, the data set name, and the disk volume serial number match the current data set being considered for IXR processing. Once an entry has been matched, no more table entries are scanned. That matched entry indicates whether the data set is to be excluded from IXR processing or processed normally. If the entire table is scanned and no match is made, the data set is processed by IXR.

In the $IXRUDSX module at label **$IXRUDST**, is the beginning of the process table. The table is built by coding the #PTBL macro for each entry in the table and also to designate the beginning and the end of the table. The #PTBL macro has the following keyword parameters:

Keyword	Meaning of Keyword
TYPE=	Indicates the type of table entry. The beginning of the table is marked with a TYPE=BEGIN and the end of the table is marked with a TYPE=END. All the table entries in between must be TYPE=ENTRY. If TYPE= is not coded, TYPE=ENTRY is the default.
FROM=	(Optional) For TYPE=ENTRY, this specifies the intercept environment to which this entry applies. Each entry in the table can be set to apply to only specific types of requesting address spaces. Possible values for FROM= are: **TSO** Specifies the entry applies only if the requesting address space is a TSO user address space. **JOB** Specifies the entry applies only if the requesting address space is a batch job. **STC** Specifies the entry applies only if the requesting address space is a started task. **JES** Specifies the entry applies only if the requesting address space is JES rescheduling a job. **ANY** (Default) Means the entry applies to every requesting address space. Combinations of the above are allowed. For example, if you wanted a given entry to apply to both TSO users and started tasks, code: **FROM=(TSO,STC)**
DSN=	(Optional) For TYPE=ENTRY, this specifies the data set name pattern mask.
VOL=	(Optional) For TYPE=ENTRY, this specifies the volume serial number pattern mask. **Note:** Some IXR intercepts do not yet have a volume serial number to associate with the data set when the call to $IXRUDSX is made. For these calls, the volume serial number is blank.
JOB=	(Optional) For TYPE=ENTRY, this specifies the job name pattern mask.
ACTION=	(Required) For TYPE=ENTRY, specifes the action to be taken by IXR if the table entry applies. The possible values are: **EXCLUDE** IXR does not process this data set. Control is returned to the user as if IXR was not active in the system for this data set. **PROCESS** IXR stops scanning the list and perform reloadability analysis, reloading the data set if necessary. This is useful for coding exceptions to the general rules.

How $IXRUDSX Uses The Process Table

The table is scanned for a match. For each entry in the table, $IXRUDSX determines whether the entry applies to the current requesting address space based on the FROM= parameter. If the entry applies, the other parameters are checked for a match. An entry is considered matched when one of the following conditions is met:

1. When only DSN= is coded and the dsn pattern mask matches
2. When only VOL= is coded and the volser pattern mask matches
3. When only JOB= is coded and the job name pattern mask matches
4. When combinations of DSN=, VOL=, and JOB= are coded and they ***all*** match

Note: The DSN=, VOL=, and JOB= parameters allow the use of CA-ASM2 pattern masking characters. See the section titled "Pattern Masking" in the *CA-ASM2 System Reference Guide* for a complete description of the pattern masking characters and how they are used.

Once an entry has been matched, no more table entries are scanned. For the entries that match, the ACTION= parameter indicates what action is taken for this data set. If ACTION=EXCLUDE was specified, IXR does not process this data set at all. If ACTION=PROCESS was specified, IXR stops scanning the table and processes this data set normally. If the entire table is scanned and no match is found, $IXRUDSX treats this as ACTION=PROCESS.

Example 1:

```
#PTBL JOB='JES3',ACTION=PROCESS
```

Any data set being processed by the JES3 converter-interpreter is processed by IXR.

Example 2:

```
#PTBL TYPE=ENTRY,DSN='-.PROJDEFS.LOAD',ACTION=EXCLUDE,FROM=(TSO,STC)
```

Any data sets whose name ends with .PROJDEFS.LOAD are excluded from IXR processing if the request is coming from a TSO address space or a started task.

Example 3:

```
#PTBL DSN='-.PROJDEFS.LOAD',FROM=TSO,JOB='SYS-',ACTION=PROCESS
#PTBL DSN='-.PROJDEFS.LOAD',FROM=TSO,ACTION=EXCLUDE
```

This example is similar to example 2 above, except it provides for some exceptions to the rule. This sequence of #PTBL entries cause IXR to reload data sets ending with .PROJDEFS.LOAD if they are required by TSO users whose TSO logon IDs begin with SYS (such as SYSTP01 or SYSOPER). The second #PTBL macro with FROM=TSO means that IXR excludes these data sets only if the requesting address space is a TSO user address space. All other types of address spaces are processed.

Example 4:

```
#PTBL TYPE=ENTRY,JOB=PROD-,ACTION=PROCESS
```

Any data set that is being referenced by a job whose name begins with PROD is considered for IXR processing.

Note: Review the #PTBL macro coding that is already set up in the first part of the $IXRUDSX module for a working example.

Testing Your $IXRUDSX IXR User Exit With PGM=TESTUDSX

The test driver program TESTUDSX can call the $IXRUDSX IXR user exit, passing it a list of user specified data set names. For each data set name given, the test program invokes the $IXRUDSX exit and report back the proposed action that IXR would take if the user exit were installed.

The SYSIN DD is where you specify the test data to be passed to the $IXRUDSX user exit. The data must be column aligned as follows:

Column	Data To Be Specified In That Column
1	The fully qualified 44 character data set name.
46	(Optional) The six character disk volume serial number.
53	(Optional) The one character address space type indicator: **B** (Default) Batch job **T** TSO user **S** Started task **R** JES restart
55	(Optional) The job name or TSO user ID. If this is left blank, the TESTUDSX program supplies JOBNM to the user exit as the job name.

After assembling the $IXRUDSX user exit with any changes to the #PTBL table, run the following JCL:

```
//TESTUDSX  JOB (account),'name',CLASS=A,
//      MSGCLASS=A
//TESTUDSX  EXEC  PGM=TESTUDSX
//STEPLIB  DD  DSN=cai.cailib,DISP=SHR     <---- CA-ASM2 Load Lib
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
JOEUSER.ISR00001.BACKUP                     WORK01 B JOEUSER
JOEUSER.ISR00001.BACKUP                     WORK01 T JOEUSER
JOEUSER.ISR00001.BACKUP                     WORK01 S JOEUSER
JOEUSER.ISR00001.BACKUP                     WORK01 R JOEUSER
JOEUSER.ISR00001.BACKUP                     SJ0005 B JOEUSER
JOEUSER.ISR00001.BACKUP                     SJ0005 T JOEUSER
JOEUSER.ISR00001.BACKUP                     PROD01 T JES3
JOEUSER.PROJDEFS.LOAD                       MVXE76 T JOEUSER
JOEUSER.PROJDEFS.LOAD                       SJ0006 T JOEUSER
JOEUSER.PROJDEFS.LOAD                       M80005 T JOEUSER
JOEUSER.PROJDEFS.LOADLIB                    SJ0008 T JOEUSER
JOEUSER.PROJDEFS.LOAD                       MVXE76 T JOEUSER
JOEUSER.PROJDEFS.LOAD                       SJ0006 T JOEUSER
```

4.15.3 $IXRUROX (IXR Reload Options Exit)

$IXRUROX (IXR Reload Options Exit) is invoked for each IXR request after IXR determines all processing options for the request and before the Data Set Selection Exit executes for any data sets associated with the request. This includes requests that are not processed unless the options are modified; for example, requests where IXR master on/off is off or the corresponding intercept on/off option bit is off. This exit examines and possibly modifies the IXR request processing options that are used.

Part of the input to the exit is a 12-byte area that represents the processing option settings determined after system-wide defaults, user option DD statements, and system-wide overrides are merged. The exit can modify the options in any way it wants. This makes it possible, for example, to deny the request altogether (by resetting the IXR processing on/off bit), to forcibly set or reset options on the basis of job name or other localized criteria, or to provide intelligence in establishing the IXR request processing priority.

The first three bytes of the option field contain the on/off-type options, each represented as a single bit. If a bit is on, the corresponding option is regarded as on, yes, or activated. If a bit is off, the option is treated as off, no, or not activated. Six of the remaining nine option bytes contain the IXR processing priorities corresponding to intercept IDs 1-6, respectively. The 12-byte option area mapping is contained in a control block called $OPT. This is distributed in CAI.CAIMAC as macro #OPT. A content description of the #OPT area is shown following.

Field Name	Structure	Description
$OPT	DSECT	Name of mapping DSECT
$OPTX	0XL12	Entire option string
$OPXBITS	0BL3	IXR "bit" options
$OPX1	BL1 --	First byte of bit options --
$OPX1IXR	X'80'	-- IXR processing on/off
$OPX1UJI	X'40'	-- Job staging intercept on/off
$OPX1USI	X'20'	-- Step staging intercept on/off
$OPX1S99	X'10'	-- Dynalloc intercept on/off
$OPX126A	X'08'	-- Allocation LOCATE intercepton/off
$OPX126V	X'04'	-- VSAM OPEN intercept on/off
$OPX1213	X'02'	-- Non-VSAM OPEN intercept on/off
$OPX1DFR	X'01'	-- Defer reloads until OPEN yes/no
$OPX2	BL1 --	Second byte of bit options --
$OPX2	BL1 --	Second byte of bit options --

Field Name	Structure	Description
$OPX2INT	X'40'	-- Interactive mode on/off
$OPX2QUI	X'20'	-- Quiet mode on/off
$OPX2522	X'10'	-- S522 abend prevention on/off
$OPX2CND	X'08'	-- Conditional staging on/off
$OPX2ASY	X'04'	-- Automatic asynchronous processing on/off
$OPX2IAS	X'02'	-- Interactive ASYNC processing permitted yes/no
$OPX2SEC	X'01'	-- Secondary archive reload permitted yes/no
$OPX3	BL1 --	Third byte of bit options --
$OPX3FRC	X'80'	-- Force reload yes/no
$OPX3VER	X'40'	-- Verbose mode on/off
$OPX3JS3	X'20'	-- JES3 support on/off
$OPX3X37	X'10'	-- STOPX37 Assist on/off
...	X'0F'	-- Reserved.
$OPXBYTS	0XL9	Byte-size options
$OPXBPTY	0XL6	Intercept request priorities
$OPXBJIP	FL1	Job staging request priority
$OPXBSIP	FL1	Step staging request priority
$OPXB99P	FL1	Dynalloc request priority
$OPXB2AP	FL1	Allocation LOCATE request priority
$OPXB2VP	FL1	VSAM OPEN request priority
$OPXB13P	FL1	Non-VSAM OPEN request priority
$OPXBPRV	FL1	IXR privilege class
$OPXBCLS	CL1	Rechedule job wait class
$OPXBJS3	XL1	SVC26/JES-3 request priority

$OPT Control Block Mapping

The Reload Options Exit receives control in task mode, supervisor state, and the protect key of the IXR subsystem. All passed parameters are in subsystem-key storage, and thus are accessible. Register contents on entry to the exit are as follows:

Register	Contents
R0	Indeterminate
R1	Address of 12-byte parameter list: Offset Length Description 0 4 Address of $UIB control block (describes job making reload request) 4 4 Address if 512-byte work area that exit can use 8 4 Address of the 12-byte option string for the reload request
R2-R12	Indeterminate
R13	Address of a conventional 72-byte save area
R14	Return address
R15	Address of exit's entry point

The exit must save and restore any of registers 2-14 that are used. The save area addressed by R13 on entry can be used for this purpose. Registers 0, 1, and 15 need not be saved and restored. Unless the exit requires more than 512 bytes of work storage, the exit can use the work area passed as the second parameter instead of issuing MVS GETMAIN and FREEMAIN macros. The work area is not guaranteed to be all zeros when the exit is invoked, so any fields used as counters, flags, and so forth, should be zeroed by the exit before use. Code the exit to take no action when passed a request with the job name JES3. This means that JES3 CI (Converter-Interpreter), not IXR, currently controls the search for data sets.

The Reload Options Exit can examine and, if desired, modify the option string passed as the third parameter item. Any return code passed back to IXR in register 15 is ignored. If the exit wants to delete the entire request, all it has to do is reset (zero) the first bit in the option string, which is the IXR master on/off flag.

4.15.4 $IXRUSVX (IXR Volume Selection)

IXR invokes $IXRUSVX when catalog-type references are made to archived, cataloged non-VSAM data sets (for example $IXRUSVX is not given control for VOL=serial referenced data sets). The exit can examine and modify the $OPTIONS volume selection control fields that are ultimately employed for reload volume selection for the particular reload involved.

The Volume Selection Exit receives control in task mode, supervisor state, and the protect key of the IXR subsystem. All passed parameters are in subsystem-key storage, and thus are accessible. Register contents on entry to the exit are shown on the following page.

<table>
<tr><th>Register</th><th>Contents</th></tr>
<tr><td>R0</td><td>Indeterminate</td></tr>
<tr><td>R1</td><td>Address of 28-byte parameter list:

<table>
<tr><th>Offset</th><th>Length</th><th>Description</th></tr>
<tr><td>0</td><td>4</td><td>Address of $UIB control block (describes job that requested data set)</td></tr>
<tr><td>4</td><td>4</td><td>Address if 512-byte work area that exit can use</td></tr>
<tr><td>8</td><td>4</td><td>Address of $UDB control block (describes data set that was requested)</td></tr>
<tr><td>12</td><td>4</td><td>Address of $OPTIONS parameters for reload volume selection</td></tr>
<tr><td>16</td><td>4</td><td>Address of copy of an archive catalog record for data set</td></tr>
<tr><td>20</td><td>4</td><td>Address of copy of an archive catalog extension record, or zero if none</td></tr>
<tr><td>24</td><td>4</td><td>Address of a 120-byte volser list return area</td></tr>
<tr><td>28</td><td>4</td><td>Address of a 122-byte list of home volume serial numbers for multivolume data sets</td></tr>
</table></td></tr>
<tr><td>R2-R12</td><td>Indeterminate</td></tr>
<tr><td>R13</td><td>Address of a conventional 72-byte save area</td></tr>
<tr><td>R14</td><td>Return address</td></tr>
<tr><td>R15</td><td>Address of exit's entry point</td></tr>
</table>

The exit must save and restore any of registers 2-14 that are used. The save area addressed by R13 on entry may be used for this purpose. Registers 0, 1, and 15 need not be saved and restored. There are exceptions to R15, noted in the possible actions described next. Unless the exit requires more than 512 bytes of work storage, the work area passed as the second parameter can be used instead of having to issue MVS GETMAIN and FREEMAIN macros. The work area is not guaranteed to be all zeros when the exit is invoked, so any fields used as counters, flags, and so forth, should be zeroed by the exit before use. Code the exit to take no action when passed a request with the job name JES3. This means that JES3 Converter-Interpreter, not IXR, currently controls the search for data sets.

The exit has essentially four possible actions. It can:

- Do nothing, allowing the $OPTIONS-specified volume selection technique to proceed. The six fields in $OPTIONS that determine reload volume selection are $SOFTWAR (bit X'80'), $SELDISK, $HOMEVOL, $RLDGRP, $TYPEREQ, and $VOLGRP. For detailed descriptions of these fields, refer to the $OPTIONS described in Chapter 2.
- Select a specific DASD volume or set of volumes from among any that are online to the system, and return them in the volser return area whose address is the sixth parameter list item. The 120-byte area is binary zeros on entry to the exit. If it is nonzero and R15=0 when the exit returns, IXR assumes that it contains a list of reload volsers for a multivolume reload. If one or more of the specified volumes is not online to the system, the associated reload fails with return code 12, reason code 144.
- Select a generic one- to eight-character unit name and return it in the volser return area whose address is the sixth parameter list item. The unit name must be in the first eight characters of the volser return area, and R15 must be set to 4.
- Modify the $OPTIONS-specified volume selection criteria whose address is passed as the fourth parameter list item. The format of the selection criteria fields is documented in the basic $OPTIONS source member distributed with CA-ASM2.

4.16 $MIGDEVT (Migrate Device Table)

This exit allows you to control values used by $MIGRATE in carrying out the OPTBLOCK and REBLOCK functions by coding block sizes in the Migrate Device Table. A default table is provided with generally acceptable block sizes coded for all supported device types.

4.17 $MIGEXIT (Migration Exit)

$MIGRATE invokes the exit routine $MIGEXIT at the start and end of each input volume, when a data set is tentatively selected, after a data set is copied, and at the end of the job. For partitioned data sets, the exit is also invoked after the directory is read. A flag indicates the nature of each call. $MIGEXIT must be link edited as authorized.

The following examples are ways you can tailor the exit routine:

- Use other than age as a criterion for migration.
- Alter the space requests for output data sets.
- Select a particular output volume for each data set.
- Rename output data sets.
- Leave input data sets on the source volumes (that is, copying rather than migrating).

Information is passed to the exit routine in a shared data area ($MIGAREA). You should use an assembly listing (not a source listing) of $MIGEXIT for reference.

The EXITFLAG byte has one (and only one) bit set to show the purpose of each call; the bit values are defined in $MIGAREA. Your exit routine must test this flag on entry and ignore irrelevant calls.

Exit Routine Linkage: When $MIGRATE is invoked by an EXEC statement, it loads the exit routine from the job or step libraries at the beginning of the job, and deletes it at the end of the job. Entry to the exit routine is by a CALL macro. If you invoke $MIGRATE from a program, you can load your exit routine and pass its address to $MIGRATE. In this case, $MIGRATE does not attempt to delete it and it need not be named $MIGEXIT.

The shared data area contains four fullwords that are zeroed before the first call to the exit routine and are not changed thereafter. You can use them for any purpose in your exit routine. If you invoked $MIGRATE from a program, an additional fullword of information is passed from your calling program to the exit routine.

$MIGEXIT can be used to specify or override SMS class specifications. Fields SMSSCLS, SMSMCLS and SMSDCLS in $MIGAREA were added in Version 4.0 for this purpose.

A brief summary of the types of exit calls and their uses follows. See an assembly listing of $MIGEXIT for a detailed listing.

Start-of-Job Call (EXITFLAG=$EXSTRT)

This call is made once only at the start of processing after the PARM has been examined but before any other processing. The call is suitable for the user exit to do initialization processing such as opening its own files or dynamic allocation of

SYSUT1xx or SYSUT2xx volume DD statements. The exit may terminate the run without any further processing by setting byte MIGFLAG to any nonzero value. (This is zero when the exit is called.)

Start-of-Volume Call (EXITFLAG=$EXOPEN)

This call is made at the beginning of each input volume. You can use it to initialize your routine, inspect and modify $MIGRATE options, print messages on $MIGRATE's message data sets, and select an output volume. You can also pass information to your exit routine in the PARM field, as this field is available to the exit.

End-of-Volume Call (EXITFLAG=$EXEOV)

This call is made at the end of each input volume and has the same information available as the start-of-volume call. A typical use is to issue your own summary messages for the volume.

End-of-Job Call (EXITFLAG=$EXEOJ)

This call is made at the end of a $MIGRATE run. The only things $MIGRATE does after this call are (1) issue space use statistics for the output volumes, (2) issue a closing message, and (3) delete the exit routine (if it was loaded by $MIGRATE). You can use this call for any end-of-job cleanup required by your routines.

Data Set Call (EXITFLAG=$EXDS)

This call is made each time $MIGRATE selects a data set. If the ALL option is used, this call is made for all sequential and partitioned data sets on each volume. The following fields are available:

- Format-1 DSCB for the data set
- Input volser
- Tracks allocated, tracks used, secondary allocation
- $MIGRATE option flags
- List of output volumes
- Parameter list and work area for a LOCATE macro (catalog lookup)

The catalog-checking logic for the ONLYCAT option is in the exit, so you can modify it to meet special requirements.

You can use this exit call to:

- Change the output MSVGP name (if facility is in use).
- Select a data set for migration based on your own criteria.
- Alter space allocations for a data set.
- Choose an output block size.
- Choose an output volume.
- Rename the copied data set.

- Catalog, uncatalog, or scratch the data set.
- Write messages on the $MIGRATE message data sets, or elsewhere.

PDS Call (EXITFLAG=$EXPDS)

This call is made for a partitioned data set that is selected for migration (after the data set call) and after the directory of the data set is read into main storage. In addition to the information available in the data set call, the following are available:

- Number of directory blocks allocated.
- Number of directory blocks used.
- Contents of the directory itself. All member entries are chained together in collating sequence; alias entries are chained to their corresponding member entries. See the DSECTs PDSTABLE and PDSDIR for layout of this area.

You can use this exit call to:

- Alter the directory size.
- Select for migration based on directory information.
- Select an output volume based on directory information (for example, separate load modules from source libraries).

End-of-Data-Set Exit (EXITFLAG=$EXEDS)

This call is made after an attempt to copy a data set and before $MIGRATE checks the results. In addition to the information available in the data set call, the event control block (ECB) for the data mover is available so you can inspect the return or completion code.

You can use this exit call to do any end-of-data-set processing required by your program.

Writing Your Own $MIGEXIT

To make your own version of $MIGEXIT, modify the source code supplied in CAI.CAISRC and assemble it to produce a new $MIGEXIT. When assembling, you must include CAI.CAIMAC in your assembler SYSLIB list, as it contains necessary macros and COPY code. The resultant load module must either replace the original one in CAI.CAILIB, or be placed in a separate library that appears ahead of CAI.CAILIB on concatenated DD statements. However, if you invoke $MIGRATE from a program, you can load and then supply a pointer to the exit routine. Its name and location are then up to you.

You can write messages to the SYSPRINT or SYSTERM data sets, or to both at once, by using the SYSPRINT macro instruction:

```
name    SYSPRINT message,T=type
```

where:

```
message   is the symbol naming the message text which must have a correct
          length attribute.
```

type is the type of message to issue. If omitted, the message is written to SYSPRINT only. If T=T is specified, the message is written to SYSTERM only; if T=B is specified, the message is written to both SYSPRINT and SYSTERM.

For example:

```
       SYSPRINT MSG1,T=B
MSG1  DC  C' HELLO FROM $MIGRATE USER EXIT'
```

writes:

```
HELLO FROM $MIGRATE USER EXIT
```

on SYSPRINT and SYSTERM.

Note: The message text must include the carriage control byte; ANSI control is used.

When you want to construct a message with several fields, you can use the MSG and MSGEND macros:

```
MSG2      MSG       '1ON VOLUME '
MSVOL     DC        CL6' ',C' THERE WERE'
MSNDS     DC        CL6' ',C' DATA SETS MIGRATED'
          MSGEND
```

In the previous example, the length of MSG2 includes all of the lines shown.

4.17.1 Selecting Output Volumes

$MIGRATE constructs a control block for each output volume and builds a circular chain of these control blocks. Normally, $MIGRATE assigns output volumes in a round-robin fashion to the various output volumes. If there is not enough space on a volume for a particular data set, it tries the next volume. A record is kept of the smallest data set that could not be allocated on each volume.

If a one-track data set cannot be allocated on a volume or if allocation fails because of a VTOC I/O error or no free DSCBs, the volume is marked unusable by setting the smallest space that could not be allocated (MINFAIL) to zero. $MIGRATE does not attempt to allocate a data set that is larger than MINFAIL on a volume.

You can inspect the volume list and data set information available in the data set and PDS exits, and make your own selection of an output volume. Relevant variables are:

@OVOLS Contains the address of the first control block in the list.

#OVOLS Contains the number of entries in the list (fullword integer).

WRKPTR Points to the current entry.

@OVOL Is zero upon entry to the exit routine. You can place the address of a volume entry in @OVOL, in which case $MIGRATE uses the specified volume. If allocation fails on this volume, no attempt to retry is made.

The volume control block is described by the DSECT DEFWRK.

4.17.2 Renaming and Copying Data Sets

You can cause the migrated (output) data set to have a different DSNAME than the input data set. You can also cause data sets to be copied without the input data sets being scratched.

You can make backup or test copies of data sets by changing the names in some orderly way, for example, by inserting a new index level in the name and cataloging the output data sets leaving the input unchanged.

MIGFLAG is a flag, set by the exit, to specify any combination of the actions shown on the following page.

Value	Meaning
$MIGCOPY	Copy the input data set. If this bit is off, no action is taken for this data set.
$MIGRCAT	Recatalog the data set after a successful copy. Should not be set if the input data set is not cataloged,
$MIGSCR	Scratch the input data set after a successful copy (and successful recataloging, if specified). If not set, the input data set is left on disk after the copy.
$MIGCAT	Catalog the output data set after a successful copy. Must not be used unless either (1) the input data set is not cataloged or (2) the data set is renamed.
$MIGUCAT	Uncatalog the input data set after a successful copy. This option is useful in conjunction with the $MIGCAT, $MIGSCR, and rename options.
@DSCB	Points to the Format-1 DSCB of the input data set.
@NEWDSN	Points to a 44-byte field containing the DSNAME of the input data set. To rename the data set, change this field to the new name (do not change the pointer).

In the standard exit, these values are set as follows. First, the catalog is checked to see if the data set is cataloged on the source volume. If it is, the $MIGCOPY, $MIGRCAT, and $MIGSCR bits are set. If it is not, no action is taken if ONLYCAT is set. If it is not and NOONLYCAT is set, the $MIGCOPY and $MIGSCR bits are set.

4.17.3 Catalog Checking

The exit routine issues a LOCATE macro to check whether the input data set is cataloged. If your data sets are always cataloged (or never cataloged), you can delete this and reduce overhead.

4.17.4 Space Allocation

The number of tracks allocated and used by the input data set are stored in INPALC and INPUSED, respectively. The output primary is set to INPALC unless ALLOUSED was specified as a PARM value, in which case the output primary allocation is equal to INPUSED. If the output device is unlike the input device, the value is converted to allow for the different track length. You can change INPALC or INPUSED if you want to change the size of the data set. You can also change INPSEC if you want to change the secondary allocation value.

The exit contains code to set a minimum secondary allocation value of one-eighth the primary allocation (minimum of one). This provides a reasonable allowance for those cases where space conversion is not exact. In general, it is not possible to compute a precise space allocation value when data sets are copied to unlike devices, except for data sets with fixed-length (RECFM=F or FB) records. Computer Associates recommends that this minimum secondary not be reduced unless you are copying between like devices.

CAI.CAISRC contains the following members of use in writing a $MIGEXIT:

Member	Type	Purpose
$MIGAREA	DSECT	Data area shared by user exit and $MIGRATE. Read this for parameters that $MIGRATE can set.
$MIGEXIT	MODULE	Standard exit routine - copies data sets only if they are cataloged on the source volume. Issues error messages if the data set is not cataloged. You may issue your own messages from this exit if you want.
DEFWRK	DSECT	Definition of @OVOL entries, volume select
ENTER	MACRO	Routine initialization.
EXIT	MACRO	Routine cleanup and exit.
FILL	MACRO	Character fill.
MTESTER	MODULE	Sample driver program.
REGISTER	MACRO	Register equates.
SPVSET	COPIED	Set global variable.
SYSPRINT	MACRO	Issue messages to SYSPRINT.

The following information tells you how to do some useful things in a $MIGEXIT:

- Uncatalog the old data set after a successful copy: turn on bit $MIGUCAT of MIGFLAG.
- Scratch the old data set after a successful copy: turn on bit $MIGSCR of MIGFLAG.
- Rename the new data set: change the data pointed to by @NEWDSN.
- Catalog or recatalog the new data set after a successful copy: turn on bit $MIGCAT or $MIGRCAT of MIGFLAG.
- Control copying of data set: turn on bit $MIGCOPY of MIGFLAG.
- Change space allocation for new data set: Use SPACEFLG for setting cylinder allocation or allocating only used tracks. Change INPPRI and INPSEC for primary and secondary allocation values (these are in terms of the input volume tracks). This quantity can be converted based on the track capacity of the output device.
- Change block size for new data set: @BLKSIZE points to a fullword that contains zero at entry. Place the block size into this fullword. Do not change the pointer. Also check @MVRFLAG in $MIGAREA; it points to a bit field that can cause reblocking or track overflow to be used.
- Change the target volume: @OVOLS points to a circular list of the volume entries. #OVOLS is the number of entries. @OVOL is the pointer you can use to select a volume. The volume entries are defined in member DEFWRK.
- Modify creation or expiration dates: Modify the DSCB pointed to by @DSCB.

4.18 $NTEXIT (New Tape Exit)

$NTEXIT receives control whenever CA-ASM2 starts a new output tape volume, master or duplex. The exit has three functions:

1. Interface with tape management program products.
2. Perform any special actions required for unit allocation of the desired tape device.
3. Supply a data set name for the (dummy) first file on the output tape, as an internal tape label.

The exception to this is when a volume is dumped to a backup tape by $DEFRAG in CA-ASM2 Mode, and the EXEC statement that invokes $DEFRAG includes the parameter CATALOG. Then, the invoking JCL also supplies the file-1 data set name. In such a case, trying to duplex the tape with $NTEXIT's default naming conventions (see following) causes the duplex job to abend. To duplex a tape made this way, you must either change $NTEXIT to supply a name that matches the name on the master tape; or set up $COPYTP, CA-ASM2's tape duplexing utility, to run in Independent Mode. This means invoking it from the procedure ASM2CPYT and supplying the correct tape label with the DSN parameter in the INTAPE DD statement of the procedure. (See the topic "Tape Duplexing Utility" in Chapter 15 of the *CA-ASM2 System Reference Guide*.)

You define parameters which $NTEXIT uses for function 1 above by changing member $XTTMS in CAI.CAIMAC.

You can change the way $NTEXIT does functions 2 and 3 above.

Discussed here is the assigning of data set names to output tapes.

$COPYTP, when duplexing a standard-label tape in CA-ASM2 mode, passes control to $NTEXIT to get the name of the first data set on the master tape. $NTEXIT must reconstruct this name, knowing only the volser of the master and whether it is an archive or backup tape. This means that if you change the naming scheme in $NTEXIT, tapes that have the old names cannot be duplexed. This is because $COPYTP uses the name as reconstructed by $NTEXIT to try to open the master tape. If the name from $NTEXIT does not match the name in the first tape file, OPEN issues an abend. Pick one naming scheme to use in $NTEXIT and be consistent. Do not, for instance, change $NTEXIT with each version of CA-ASM2, making tape names version-dependent.

You can use $NTEXIT as supplied by Computer Associates without modification if your installation does not care what name is applied to the dummy file. The following constant data set names are used by the distributed $NTEXIT according to the context:

ASM2.ARCH.MASTER.TAPE
ASM2.BKUP.MASTER.TAPE
ASM2.ARCH.DUPE.TAPE
ASM2.BKUP.DUPE.TAPE
ASM2.BKUP.DEFRAG.MSTR
ASM2.BKUP.DFDU.TAPE

You can modify the source code to $NTEXIT in CAI.CAISRC($NTEXIT) to alter these constants, or use it as the basis for a true user-written $NTEXIT.

Notes:

1. CAI.CAIMAC($NTPRMSD) is the DSECT definition of the parameter fields accessed and updated by $NTEXIT.
2. The $COPYTP utility requires that:
   - The high-level index of the data set names listed above must be the prefix (if any) chosen earlier for all CA-ASM2 data sets. The default names here assume that you chose CA-ASM2.
   - The second-level index (or high-level, if you did not choose a prefix) must be ARCH or BKUP.

4.19 $PDMUEXT (Physical Data Mover Unload Exit)

This exit allows you to override LSTAR processing, the $PDM default method of dumping data sets to tape. LSTAR processing saves space by dumping only the used tracks within a data set, not the allocated but unused tracks. (LSTAR is the name of a field in the Format-1 DSCB of a PS or PO data set, containing the relative address of the last track used.)

For each data set that defaults to LSTAR processing, $DEFRAG or $PDM passes to $PDMUEXT a pointer to the Format-1 DSCB. You can code the exit to use the information in the DSCB any way you like to determine whether to override LSTAR processing for a given data set and dump everything, including unused tracks. The output from the exit must be one of two values in register 15:

R15 = 0 Tells $PDM use default (LSTAR) processing.

R15 not = 0 Tells $PDM not to use LSTAR processing, instead dump all tracks allocated within the data set.

$PDMUEXT, as installed, is a dummy module that always passes 0 to register 15.

4.20 $RELDEXT (Reload Exit)

$RELDEXT permits appropriate processing at the termination of any data set retrieval request. For example, it notifies you of the result of the retrieval request.

$RELDEXT is given control after all processing on a data set is completed. Data sets that CA-ASM2 attempted to reload and the results of retrieval requests for which there are syntax errors are passed to $RELDEXT. The further processing of CA-ASM2 is not influenced by any action performed by $RELDEXT. That is, CA-ASM2 does not inspect any return information from $RELDEXT.

See CAI.CAIMAC($RELDPRM) for the DSECT definition of input parameters passed to $RELDEXT.

4.21 $RXQEXIT (Queued Reload Exit)

The $RXQEXIT allows you to decide whether to make a reload operation proceed, and if so, whether it is to be queued or immediate.

Register 1 points to four fullwords.

Fullword	Contents
1	Points to the table entry of the data set to be reloaded
2	Address of the CPPL (Command Processor Parameter List)
3	Address of the CA-ASM2 catalog record for this data set
4	Address of the environment flags

Fields in the table entry which the exit may modify include:

SQTYPE TSO/batch indicator (T=TSO, J=batch)

SQUID User ID or batch job name

SQULEN Length of user ID

SQNAME Logical queue name to be used for selective batch queued reload

Return code in register 15 as follows:

0 Do the reload now
12 Bypass this entry, do not queue or restore
all others Put entry in queue

Message ID = $RQ

4.22 $SELVOLX (Select Volume)

This exit can be used to extend the allocate by UNITNAME capabilities of the $RLDGRP field of $OPTIONS.

$SELVOLX is invoked by $RA and IXR, and provides the option of placing one or more volsers or a UNITNAME in the USERVOL field or modifying any of the fields listed next. If you specify a volser, you must ensure that the volume has an extent large enough to contain the primary allocation of the data set. To indicate that the value in USERVOL is a UNITNAME, you must set R15 to 4 upon exit. For all other return codes USERVOL is assumed to contain one or more volsers terminated by a binary zero. This exit is invoked only if it is in the JOBLIB, STEPLIB, or link list library. If $SELDISK=3 in $OPTIONS, you must supply a volser in the exit and the exit must be available.

Upon return from the user exit, CA-ASM2 uses the modified options in volume selection.

Upon entry to $SELVOLX, register 1 points to a list of addresses as follows:

Address	Description
A(HOMEVOL)	Address of $HOMEVOL from $OPTIONS.
A(VOLGRP)	Address of $VOLGRP from $OPTIONS.
A(TYPEREQ)	Address of $TYPEREQ from $OPTIONS.
A(SELDISK)	Address of $SELDISK from $OPTIONS.
A(TARGETV)	Address of field containing NEWVOL, if used.
A(USERVOL)	Address of field where the user exit places the volser, volser list (up to 20 volumes) or UNITNAME. (R15 must be set to 4 to indicate USERVOL is a UNITNAME.) If a UNITNAME is selected, the user is responsible for ensuring that the UNITNAME group has a sufficient number of volumes and space on the volumes for the reload.
A(ASM2CAT)	Address of CA-ASM2 catalog record.
A(UNLVOLS)	Address of a list of up to 20 volsers the data set resided on at unload time, preceded by a 2-byte binary volser count. Volsers must be six characters with no blanks or commas between entries in the list.

4.23 $TSINQ (Tape System Inquiry)

This module is invoked by CA-ASM2 routines needing to determine if a tape being externally managed by CA-1 or TLMS has expired. It must be assembled by the user after the $XTTMS member of CAI.CAIMAC has been set up to reflect the user's tape management system. The tape system is invoked to determine if the input tape requested has expired or not.

The following return codes are issued by this routine:

8	Tape has expired
12	Unable to open or process tape catalog
16	A tape system has not been installed ($XTTMS parm &TAPEMGT is null)

Depending on the settings of various assembly variables in source member $XTTMS, this exit can be assembled to support:

No tape management system (the default)

TMS for versions earlier than 5

TMS5 for Version 5.1 or later

TLMS - TMLS users must modify this exit to include correct VMFBASE record. See label TLMSBASE in the provided source for further instructions.

Further discussion can be found in member $XTTMS. Installations should review this member before assembling this program. In addition, source members $NTEXIT and $BTEXIT should be reviewed for applicability.

In addition to the assembly variations that result from the settings in $XTTMS, the installation is permitted to modify this exit as required.

4.24 $UVEXIT (Update Retrieval)

$UVEXIT allows you to update (with your installation specified data) the DSCB of a newly retrieved data set at the conclusion of a successful retrieval.

$UVEXIT receives control after all reload operations are performed and after all CA-ASM2 modifications to the Format-1 DSCB are made. Upon entry to $UVEXIT, register 1 points to a parameter list consisting of two addresses. The first address is that of the DSCB; the second address is that of the volser being processed. If $UVEXIT changes the in-core DSCB, CA-ASM2 updates the DSCB on disk.

See CAI.CAIMAC($UVPARM) for the DSECT definition of the input parameters passed by CA-ASM2 to $UVEXIT.

4.25 $VLISTXT (VLIST Utility Exit)

The $VLISTXT exit allows you to control the use of $VLIST. It is called by $VLIST three times.

The first call is a preprocessing call which can be used to control the use of $VLIST. A return code other than zero causes $VLIST to end execution.

The second call is the catalog process call. This is performed for every $VLIST control statement read with SYSIN and can be used to limit access to catalogs. A return code other than zero causes $VLIST to skip the catalog control card just read.

The last call is the postprocessing call which is made after $VLIST has finished processing and is about to terminate. This call can be used to perform housekeeping (i.e., FREEMAIN of acquired areas) which may be required as a result of processing performed in the preprocessing or catalog process calls.

At entry, R1 points to the $VLIST user exit parameter list which is mapped out as follows:

```
R1 ====>    Offset    Length     Description

            +0        8          Type of call (see list following)
            +8        4          Address of $VLIST parms
            +12       4          Address of 44-byte catalog name
            +16       4          Address of catalog's password (8 bytes)
            +20       4          Address of error message file DCB
            +24       4          Address of 256-byte user work area

            Layout of $VLIST parms

            +0        2          Halfword length of $VLIST parms
            +2        3          Three-character parameter entry
```

Type of call

```
     PRE      - Preprocessing call
     PROCESS  - Catalog processing call
     POST     - Postprocessing call
```

4.26 $VSAMRXT (VSAM Rename Exit)

The $VSAMRXT exit allows you to change the names generated by $VSAMIOR for the data and index components of a cluster which is being renamed at reload. As a default, $VSAMIOR generates the following names:

Cluster	Name specified on the NEWNAME keyword of the $RA/$RB command
Data Component	NEWNAME+.DATA or NEWNAME+.D
Index Component	NEWNAME+.INDEX or NEWNAME+.I

When NEWNAME is 38 characters or less, .DATA or .INDEX is used.

The abbreviation .D or .I is used when NEWNAME is greater than 38 characters but less than 43 characters.

If NEWNAME is equal to or greater than 43 characters .D or .I are made the last index level of the data set name.

Upon entry into $VSAMRXT, R1 contains the address of a VSAM Component Table. The CAIMAC member M2$VSC contains a DSECT that maps this parameter area. The VSC#ASO field contains the number of entries in the list. In the case of reloading a VSAM sphere with NEWQUAL( ), this list includes data, index, path, alternate index and cluster names. The format of the table is as follows:

```
R1 ====>    Offset    Length     Description

            +0        1          Component entry type
                                 "D" = data component
                                 "I" = index component
                                 " " = end of entries
            +1        44         Component entry name
            +45       44         Name generated by $VSAMIOR
            +89       1          Unused
```

This exit can change or completely replace the $VSAMIOR generated name at offset +45 of any entry. It should not modify the contents of offsets 0-44.

4.27 AMCUSREX (Allocation Manager User Exit)

The Allocation Manager user exit is given control:

- Immediately following the job type determination (job type processing).
- Immediately after the data set type and action to be taken are selected (selected data set processing), or
- Immediately after it is determined that the data set is not to be processed by Allocation Manager (nonselected data set processing), and
- After all JCL for the job has been processed (job ENQ processing).

The user exit can:

- Command the unit not be overridden.
- Modify the job type, generic name override, data set type, SIOT or JFCB.
- Allow the job to be processed or fail the job as a JCL error.

A sample Allocation Manager user exit is distributed in source form as member AMCUSREX in CAI.CAISRC. This exit includes code to:

- Allocate permanent data sets to specific units depending on EXPDT or retention period.
- Override the UNIT name with the IBM generic name for the device if the data set is a system-generated data set used for volume allocation.

These functions are discussed in detail in this section. Dummy routines are also provided for ease in implementing additional functions as required.

Allocation Manager determines if the user exit is to be taken by locating module AMCUSREX in a linklist library. If none is found, Allocation Manager assumes no user processing is necessary.

AMCUSREX gets control in supervisor state, protect key 0 with AMODE 24 and RMODE 24. The exit must be reentrant.

Note: A second version of AMCUSREX is distributed under the name AMCUSRX2 in CAI.CAISRC. This sample AMCUSREX exit may be installed to identify and optionally fail jobs using JOBCAT/STEPCAT DD statements (which are not to be used in an SMS environment). It must be installed in a linklist data set as AMCUSREX.

4.27.1 Job Type Selection Processing

After the job type has been determined, the user exit is given control with the following input parameter list. Register 1 contains the address of a list of fullword addresses:

```
A(EXIT CODE)                  *Note 1
A(USER WORK AREA)             *Note 2
A(ALLOCATION MANAGER TABLE)
A(JOB TYPE CODE)              *Note 3
A(JOB NAME)
A(ACCOUNTING INFORMATION)
A(PROGRAMMER NAME)
```

See the end of this description for parameter list note explanations. The return code in register 15 indicates the action taken by the exit.

Return Code	Action
0	No action is taken.
4	Fail the job as a JCL error.
8	Job type is replaced. If the replacement job type is not defined in the current Allocation Manager Table, the job is classified and processed as an UNKNOWN job type.

4.27.1.1 Selected Data Set Processing

After the data set type and return code are selected, the user exit is again given control. The input parameters pointed to by register 1 are:

```
A(EXIT CODE)                *Note 1
A(USER WORK AREA)           *Note 2
A(ALLOCATION MANAGER TABLE)
A(JOB TYPE CODE)            *Note 3
A(JOB NAME)
A(ACCOUNTING INFORMATION)
A(PROGRAMMER NAME)
A(ACTION CODE)              *Note 4
A(DATA SET TYPE CODE)       *Note 5
A(GENERIC NAME OVERRIDE)
A(UNIT NAME CODED ON DD)
A(DATA SET NAME)
A(SIOT)
A(JFCB)
A(SCT)
```

See the end of this description for parameter list note explanations.

The return code in register 15 indicates the action taken by the exit.

Return Code	Action
0	No action is taken.
4	Fail the job as a JCL error.
8	Do not override the UNIT name.
12	Generic name is replaced.
16	Data set type is replaced.
20	SIOT/JFCB is modified.

One of the functions in the supplied user exit allocates permanent data sets (Type 04) to specific units depending on EXPDT or retention period. This is accomplished in the ACUESDPM routine. The UNIT name to retention period relationship is defined in the DAYTABLE table entries. This table needs to be changed by the user to meet the needs of the user installation. A data set with a retention period equal to or less than the number of days in a table entry is overridden with the UNIT name in that entry. The keyword generic name ALLOW in the entry suppresses the override. This action is to be taken only for:

- Permanent data sets not identified by the keyword expiration date (33nnn).
- Permanent data sets that are not changed (action code X'00').
- Permanent data sets that are to be overridden (action code X'01') by the Allocation Manager criteria.

4.27.1.2 Nonselected Data Set Processing

After the data set type and return code are selected, the user exit is again given control. If the data set is not to be processed by Allocation Manager (non-DASD data sets), the user exit is given control to perform any checks that may be required. The input parameters are the same as for the selected data set. However, the action code and data set type code are zeros and the generic name override is equal to the UNIT name from the DD statement. On certain types of DD statements (for example, SYSOUT, DD *, DD DUMMY) the operating system supplies a UNIT name for the data set. This UNIT name is passed to the user exit in two parameters: GENERIC NAME OVERRIDE and UNIT NAME CODED ON DD.

The return code in register 15 indicates the action taken by the exit.

Return Code	Action
0	No action is taken.
4	Fail the job as a JCL error.
8	No action is taken.
12	Generic name is replaced.
16	Invalid, Job is failed.
20	SIOT/JFCB is modified.

One of the functions in the supplied user exit determines if a data set is a system-generated data set used for volume allocation. If it is volume allocation, the UNIT name is overridden by the IBM generic name for the device. The routine that

performs this function is ACUEDSNS. The device type codes and associated names are defined in the UCBDEVIC table.

4.27.2 Job ENQ Processing

After all JCL has been processed for a particular job, the decision is made to fail the job or allow processing. The user exit is once again given control with the following input parameters pointed to by register 1:

```
A(EXIT CODE)                    *Note 1
A(USER WORK AREA)               *Note 2
A(ALLOCATION MANAGER TABLE)
A(JOB TYPE CODE)                *Note 3
A(JOB NAME)
A(ACCOUNTING INFORMATION)
A(PROGRAMMER NAME)
A(ACTION CODE)                  *Note 4
```

Refer to the end of this description for parameter list note explanations.

The return code in register 15 indicates the action taken by the exit.

Return Code	Action
0	No action is taken.
4	Fail the job as a JCL error.
8	Allow the job to be processed.

4.27.3 Parameter List Notes

Note 1

The EXIT CODE indicates which processing exit is being taken.

X'01' - Job type selection
X'02' - Selected data set processing
X'03' - Nonselected data set processing
X'04' - Job ENQ processing

Note 2

The user exit module must be reentrant and reusable. For performance reasons and ease of use, the USER WORK AREA points to a 64-byte work area. This work area is initialized to binary zeros by Allocation Manager at job type selection time.

Note 3

The JOB TYPE CODE indicates what job type has been selected. It is a 2-byte field. Where byte 1 is the job type class:

X'01' - Production job
X'02' - Test job
X'03' - Systems job
X'04' - Unknown

Where byte 2 is the job type subclass:

A hexadecimal number from '00' to 'FF'

Note 4

The ACTION CODE indicates the action being taken. When selected data set processing equals

X'00' - No action taken
X'01' - UNIT name overridden
X'02' - UNIT name allowed
X'04' - DD statement in error

When job ENQ processing equals

X'00' - No failure
X'04' - Job fails

Note 5

The DATA SET TYPE CODE indicates the type of data set.

X'01' - Temporary
X'02' - Nontemporary without an EXPDT in the future (and not 33nnn)
X'03' - Generation data group member
X'04' - Nontemporary with an EXPDT in the future (or 33nnn)

Appendix A. Worksheets

This appendix contains all of the worksheets that are to be filled out with your selections before starting the installation of CA-ASM2. There are a total of six worksheets:

- Base Installation
- $OPTIONS
- General Protect Criteria
- IXR Installation
- IXR Startup Parameters
- Allocation Manager Installation

After you have completed the installation, return the worksheets here for future reference.

A.1 Base Installation Worksheet

Item	Your selection	Default
1. ACCT	_____	JOBACCT
2. PROGM	_____	PROGRAMMER NAME
3. JOB2	_____	JOBCARD2
4. JOB3	_____	JOBCARD3
5. JOB4	_____	JOBCARD4
6. CAINODE	_____	CAI
7. CSINODE	_____	CAI.VSAM
8. CCLISTL	_____	CAI.CAICLIB
9. CLOADLIB	_____	CAI.CAILIB
10. CLPALIB	_____	CAI.CAILPA
11. CMACLIB	_____	CAI.CAIMAC
12. CPROCLIB	_____	CAI.CAIPROC
13. CISPFLIB	_____	CAI.CAIISPM
14. CINSTOP	_____	CAI.PPOPTION
15. CSRCLIB	_____	CAI.CAISRC
16. RELPFX	_____	CAI.ASM2.R42
17. PREFIX	_____	CAI.ASM2
18. VPREFIX	_____	CAI.ASM2V
19. CAITGT	_____	CAITGT
20. CAIDLIB	_____	CAIDLIB

A.2 Base Installation Worksheet (Continued)

Item	Your selection	Default
21. CSISER	_____	
22. SMPSER	_____	
23. SMPUNIT	_____	3390
24. TGTSER	_____	
25. TGTUNIT	_____	3390
26. DLBSER	_____	
27. DLBUNIT	_____	3390
28. WRKUNIT	_____	SYSDA
29. IPOSMPE	_____	IPOSMPE
30. MVSTGT	_____	MVSTGT
31. MVSDLB	_____	MVSDLB
32. SMPEREL	_____	4
33. SOUT	_____	*
34. TLBSER	_____	
35. TLBUNIT	_____	
36. PRCVOL	_____	
37. PRCUNIT	_____	
38. AVOL	_____	
39. AUNIT	_____	3390
40. IPCVOL	_____	

A.3 Base Installation Worksheet (Continued)

Item	Your selection	Default
41. JNLVOL	_____	
42. LMTYP	_____	JOB
43. LMOPT	_____	U
44. PTF196W	_____	
45. PTF192C	_____	
46. FMIDIDAT	_____	
47. BASE	_____	018
48. EXP	_____	000
49. PTFTE2	_____	

A.4 $OPTIONS Worksheet

$PRTVOL_____	$HOMEVOL_____
$WTOSW_____	$SELDISK_____
$TAPPOOL_____	$KEEPEXP_____
$KEEPSER_____	$SYSTIME_____
$USRTIME_____	$ULMTIME_____
$PRMTIME_____	$PLMTIME_____
$BKUTIME_____	$RELDOPT_____
$HARDWAR_____	$SOFTWAR_____
$ACSMETH_____	$DUP#MAX_____
$DUPTMAX_____	$USRAUTH_____
$MISCOPT_____	$ASM2FLG_____
$ANOCPY_____	$BNOCPY_____
$MAXCRL_____	$ASM2F2_____
$MNTOPT_____	$INTCSG_____
$MINSPL_____	$MSSMAX_____
$ARCHTKC_____	$BKUPTKC_____
$BKUPPDM_____	$INCDFLT_____
$COMLEN_____	$COMSW_____
$DSCBF#_____	$DSCBSW_____
$DSCBLF_____	$UDSCB_____
$INSTALL_____	$TYPEREQ_____
$MSSULQL_____	$MSSULQ_____
$ARCHEXP (year)_____	$ARCHEXP (day)_____
$BKUPEXP (year)_____	$BKUPEXP (day)_____

A.5 $OPTIONS Worksheet (Continued)

$VOLGRP (1)_____	$VOLGRP (2)_____
$VOLGRP (3)_____	$VOLGRP (4)_____
$PREFIX_____	$VPREFIX_____
$CVLEXP_____	$CVLFLG_____
$SMFID_____	$IXROPT_____
$SYSOUTC_____	$PDMOPT_____
$PDSUTIL_____	$DFGTIME_____
$TPUNIT (3420)_____	$TPUNIT (3480)_____
$TPUNIT (M860)_____	$TPUNIT (3480 XF)_____
$TPUNIT (exp)_____	$TPUNIT (exp)_____
$TPUNIT (exp)_____	$M860ETI_____
$RACFOPT_____	$VSAMEXM_____
$CATID_____	$RETNOPT_____
$DMERROR_____	$RETDAYS_____
$AMINVER_____	$BMINVER_____
$AMAXVER_____	$BMAXVER_____
$ISRMLIB_____	$RLDGRP (1)_____
$RLDGRP (2)_____	$RLDGRP (3)_____
$RLDGRP (4)_____	$DSAUNIT_____
$DSACLAS_____	$DSADAYS_____
$DSBUNIT_____	$DSBCLAS_____
$DSBDAYS_____	$DPREFIX_____
$CMPTYPE_____	$THRESH_____
$DOWNTO_____	$SASMOD_____

A.6 General Protect Criteria Worksheet

PFILE_____

HINDX_____

SINDX_____

GDG_____

NONTEMP_____

AGE_____

USE_____

EXPDT_____

MAXSCR_____

PCTSCR_____

CVOL_____

OVRRIDE_____

_____

_____

_____

_____

_____

_____

_____

A.7 IXR Installation Worksheet

Item	Your selection
1. FMID401	_____
2. CSECT401	_____
3. LMOD401	_____
4. FMID21SD	_____
5. FMID26	_____
6. CSECT26	_____
7. LMOD26	_____
8. USERLIB26	_____
9. FMIDX0A	_____

A.8 IXR Startup Parameters Worksheet

DSNE_____	OPTE_____
SELVE_____	OPTBD_____
OPTBA_____	OPTBO_____
OPTTD_____	OPTTA_____
OPTTO_____	OPTSD_____
OPTSA_____	OPTSO_____
CONDP_____	CMDC_____
522T_____	522I_____
XSPOLL_____	XSPMAX_____
CSA_____	ASBP_____
ASBS_____	ASBM_____
ASBW_____	XRBP_____
XRBS_____	XRBM_____
XRBW_____	DSBP_____
DSBS_____	DSBM_____
DSBW_____	DDPRE_____
PTAP_____	TTAP_____
TCUU_____	DEVTP_____
PVOL_____	CVOL_____
UCSX_____	STOPX37_____
BACKUPS_____	MAXDRLD_____
RLDTMPNM_____	

A.9 Allocation Manager Installation Worksheet

Item	Your selection
1. AMSVC	_____
2. TSOWORK	_____
3. TSOTEMP	_____
4. TSOUSER	_____
5. FMID4DO	_____

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X