Getting Started With SMS

Z/Series Expo
September, 2005
Ruth Ferziger
Ruthf@us.ibm.com

What is System Managed Storage?

System-managed storage allows the operating system to take over many storage management tasks that were previously performed manually. This is important because the cost of a storage device is more than just the initial cost of the hardware.

Other costs include the electrical requirements, floor space, and the people needed to install, monitor, and operate the devices. Removable media, such as optical and tape storage, cost less per megabyte than online storage but require additional time and resources to locate, retrieve, and mount.

After taking all the hidden costs into account, it becomes clear that controlling and managing the growth of information systems is critical if a business is to grow and expand profitably. There are many tasks associated with storage management, such as space management, backup processing, and data set allocation.

What is SMS?

The Storage Management Subsystem consists of:

MVS Subsystem (SMS): The SMS subsystem interface assumes control from MVS SP components to analyze all JCL and dynamic allocations to determine which data sets (NEW/OLD) are to managed by the Storage Management Subsystem. SMS also determines whether data sets are to reside on DASD, optical storage, or tape. SMS assumes complete control for scheduling resources required for a data set, allocation (NEW/OLD) of a data set, and disposition of a
data set at end of STEP/JOB if the data set is or should be SMS managed.

DFSMS/MVS components (for example, AMS, Catalog) and other products that create data sets also give control to SMS to determine if a new data set should be SMS managed, and if so, where it should reside (DASD, optical, or tape). SMS must be invoked to create SMS-managed data sets. SMS must also be called to delete or rename SMS-managed data sets.

The SMS subsystem interface also supports requests to retrieve or update information from/in the active configuration.

Full Function Address Space (SMAS): The SMS address space is primarily responsible for management of the current storage management policy (active configuration). It supports services to update the active configuration, communicate with other systems in SMS complex, and maintain volume statistics and statuses.

Miscellaneous Services: SMS provides services to read/update SMS control data sets, scan VTOC/VVDS entries, translate or execute ACS routines, initialize Storage Management Subsystem, process operator commands directed to SMS, analyze an SVC dump taken by SMS, and listen to status changes for SMS controlled devices.

Subsystem Perspective:

- MVS Subsystem.
- Defined in SYS1.PARMLIB(IEFSSNxx).
- SMS subsystem initialization routine is invoked by Master Scheduler Initialization during IPL.

- SMS must be initialized before JES.
- SMS initialization options contained in SYS1.PARMLIB(IGDSMSxx).
- SMS functions execute in the caller's address space as subsystem interface routines or branch entry routines and in the SMAS via MVS Cross Memory Services or internal processing tasks.

Address Space Perspective:

- Created by SMS initialization via system address space create facility (IEEMB881).
- Main function is to manage the SMS configuration.
- Uses the Communication Data Set (COMMDS) to communicate between systems.
- Operator commands to manage configuration and SMAS.
- Independently and automatically recoverable.
- SMS address space has the following properties (as defined in the program properties table):
  - Non-swappable
  - Executes in Key 5
  - System Task
  - Bypass password protection
  - Non-cancellable

SMS Design Considerations

Users create data sets. They need to communicate their data and storage management requirements to system administrators who manage that data.

Data has intrinsic requirements based on what it looks like. It also has storage management requirements such as where it resides and system connectivity. After all, it doesn't do me any good if I'm on system A and my data is connected to system B. In addition, it has management requirements for retention, backup, and migration.

The role of the storage administrator is to centralize control of storage management. The idea behind SMS was to provide the storage administrator with tools and facilities for managing external storage and also to provide monitoring and reporting of storage usage.

Unmanaged Environment

Consider the following analogy, which was provided by storage management consultant Paul Daughenbaugh.

Suppose you were flying to New York's JFK airport. At this airport, one of the busiest in the world, traffic is controlled by air traffic controllers. If it was not, then arriving and departing aircraft could come and go whenever they wished, using whichever runway they preferred. The result would be chaos and an increased probability of collisions. Would you want to travel through such an airport?

A storage environment without system-managed storage is analogous to an airport without air traffic controllers. Allocations and deletions occur with little or no control, on whichever volume the person performing the allocation happens to choose. Some volumes may be highly utilized in terms of both space and performance while others are sitting idle. In a storage environment, a collision can be said to occur when a data set allocation fails because there is no space on the volume on which the allocation was attempted.

In the example shown, VOL001 and VOL004 are heavily populated, while VOL002 and VOL003 are not. This tends to occur because users place data sets in locations where other data sets are known to exist or because there is a lot of JCL around which references specific volumes. It makes it more likely that an allocation attempted on VOL001 or VOL004 will fail. However, if the same allocation had been attempted on VOL002 or VOL003, it probably would have worked.

Managed Environment

System-managed storage addresses the problem of unbalanced volume utilization by placing device selection under the control of the system. The system does this using a policy established by the storage administrator who defines a volume pooling structure made up of storage groups. The storage administrator also writes straightforward automatic class selection (ACS) routines that define which data sets can be allocated in which storage groups. Using these ACS routines, the storage administrator can allow the system to control as much or as little allocation of storage groups as desired.

When a new allocation occurs, the system uses the automatic class selection (ACS) routines to determine a set of storage groups in which the data set is eligible to be allocated. The system then considers criteria such as space and performance to select the specific volume or volumes on which to perform the allocation.

Why Should I Use It?

There are products other than DFSMS/MVS which provide the equivalent of IBM's system-managed storage concept. However, many of these products simply function as a front end to MVS common allocation. They provide a hook between the user's allocation and MVS to perform device selection. Once they select a device or set of devices, the allocation proceeds as it would in a non-managed environment. If it succeeds, that is well and good. But if it fails, there is no mechanism to retry the allocation.

The IBM solution is superior in that it is an integral part of the allocation process. It receives control back after the allocation is attempted or performed. If the allocation succeeds, it can then proceed and issue appropriate messages. If the allocation fails, it can retry until either the allocation is successful or all possible volumes have been tried.

DFSMS/MVS provides the key to system-managed storage in an MVS environment. In a system-managed storage environment, an installation establishes policies for how hardware resources should be used, balancing available resources with user requirements for data availability, performance, and space. DFSMS/MVS implements these policies and handles many storage management tasks, freeing users from manual storage administration and making more efficient use of storage resources.

SMS User Interactions

The system operator:
a. issues SMS operator commands
b. responds to messages
c. runs recovery procedures

The system programmer:
a. Creates SMS control data sets
b. Writes and installs any necessary installation exits
c. Diagnoses and corrects problems

The storage administrator:
a. Manages space, performance, availability and devices
b. Establishes a storage management policy
c. Creates and maintains SMS constructs and ACS routines to implement that policy
d. Maintains and manages SMS control data sets and ACS routine source

End users and applications programmers:
a. Can specify data class, storage class, and/or management class on their allocations (or let the system assign them)
b. Update existing JCL and CLISTs, as needed

Some Basic Terms

SMS complex: A system or a collection of systems (or system groups) that share a common configuration including a common active control data set (ACDS) and a common communication data set (COMMDS) pair. The SMS configuration now supports up to 32 system names, system group names, or both.

An SMS complex should not span sysplexes. All of the volumes in the SMS complex should be in the same Parallel Sysplex. The cross-system sharing functions, such as VSAM record-level sharing (RLS), partitioned data set extended (PDSE) sharing, RACF security and global shared resources (GRS) serialization, only work within the scope of a single Parallel Sysplex. These functions are not supported when the
SMS complex extends beyond the Parallel Sysplex in which they are carried out.

You should not set up multiple SMS complexes sharing the same DASD. This not only requires extra work to maintain the duplicate SMS configurations but can also create problems such as running out of disk space, since one configuration cannot know about changes made to the other configuration, such as data set allocations and deletions, and storage group and volume status changes.

System name: The name of the system where an SMS operation is being performed.

System group: This is an SMS concept. A system group consists of system names within a Parallel Sysplex, excluding those systems that are individually specified in the SMS base configuration. A system group name can represent multiple systems. This allows SMS to support more than eight systems per SMS complex while retaining the existing configuration format of the configuration data set.

A system group name matches a Parallel Sysplex name and refers to all systems defined as part of the Parallel Sysplex that are:

a. Running the same SMS configuration
b. Defined in the configuration using the name of the Parallel Sysplex to which they belong (that is, system group)

c. NOT defined in the configuration by their system names.

The system group name represents all the systems in the Parallel Sysplex which are not explicitly specified in the SCDS base configuration.

The base configuration information contains installation defaults and identifies the systems to which the SMS configuration applies.

Storage groups represent the physical storage managed by SMS, whether DASD, tape or optical.

Storage classes contain performance, availability and accessibility attributes.

Management classes are sets of space management, back up management, object class transition, and aggregate back up attributes.

Data classes are lists of allocation attributes and their values. They are intended to simply what must be specified when you allocate a data set, for example via JCL.

Automatic class selection routines are procedural sets of automatic class selection language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of predefined storage groups, for data sets and objects.

System Group

SMS complex: A system or a collection of systems that share a common configuration including a common active control data set (ACDS) and a common communication data set (COMMDS) pair. The SMS configuration supports up to 32 system names, system group names, or both.

Parallel sysplex: A collection of MVS(TM) systems in a multi-system environment supported by the cross-system coupling facility (XCF).

System name: The name of the system where an SMS operation is being performed.

System group: The system names within a parallel sysplex, excluding those systems in that sysplex, if any, that are individually defined in the SCDS. This support can be used even if the XCF is not active.

In the MVS environment, a parallel sysplex is a collection of systems linked by closely coupled hardware facilities to process customer workloads. SMS system group name support allows you to specify a system group as a member of an SMS complex.

In SMS, the term system group is used instead of parallel sysplex. A system group consists of system names within a parallel sysplex, excluding those systems that are individually specified in the SMS base configuration. A system group name can represent multiple systems. This allows SMS to support more than eight systems per SMS complex while retaining the existing configuration format of the configuration data set.

A system group name matches a parallel sysplex name and refers to all systems defined as part of the parallel sysplex that are:

- Running the same SMS configuration
- Defined in the configuration using the name of the parallel sysplex to which they belong (that is, system group)
- Not defined in the configuration by their system names.

The system group name represents all the systems in the parallel sysplex which are not explicitly specified in the SCDS base configuration.

What Is an SMS Configuration?

The SMS configuration is a storage management policy established by a storage administrator which indicates how SMS-managed data sets are to be managed by the system. It consists of a number of elements which are defined in a source control data set. When the storage administrator activates a configuration, the contents of the SCDS are copied to an active control data set, as well as to storage within the SMS address space.

One of the mechanisms DFSMS/MVS provides to help automate storage management is the SMS constructs. They are defined using ISMF and are entities which are associated with data sets and objects based on an installation-defined policy. The intent behind these constructs is to separate the logical view of data from the physical view.

The logical view of data is concerned with what the data looks like and what services it requires. This view is supported by data class, storage class, and management class. The physical view is concerned with where the data actually resides. It is supported by storage group. There is also a construct called the base configuration which is not associated with individual data sets or objects. It contains both logical and physical information and applies to the system as a whole.

These constructs together with additional constructs and information comprise the SMS configuration. The additional constructs and information will not be discussed in this presentation. For more information, see the DFSMSdfp Storage Administration Reference on the DFSMS/MVS Publications page.

What is a Minimal Configuration?

Normally, when SMS is first started on a system, it runs with a null configuration -- which is just a fancy way of saying the ACDS is empty, and so in the configuration. At some point, the storage administrator will create a valid SMS configuration. To be valid, the SMS configuration must include the base configuration information (since it contains system-wide information), at least one storage class (so that data sets can be SMS-managed), at least one storage group (so that there is some place to put the SMS-managed data sets) and a storage group ACS routine (in order to have a way to associate the storage group with the SMS-managed data sets).

In order to keep things from becoming managed that should not be (and vice versa) you'll probably want to include a storage class ACS routine as well.

What is the Base Configuration Information?

There is one base configuration per SMS configuration. It identifies the systems that the SMS configuration is to manage. It also contains the following installation defaults:

For system-managed data sets that have not been assigned a management class, DFSMShsm uses the default management class for expiration, migration, and backup information. The default management class does not apply to objects.

The default management class name is not saved in the data set catalog entry, so that you can tell the difference between data sets with assigned management classes and those without them. Specifying a default management class is optional. If a data set has no management classes and a default management class is not defined, DFSMShsm uses its own defaults for the data set.

The default unit is an esoteric or generic device name, such as SYSDA or 3390, that applies to data sets that are not managed by SMS. For new data set allocations, the default unit allows users to omit the UNIT parameter from DD statements or dynamic allocation equivalent, as they can when allocating system-managed data sets. The default unit does not apply to objects.

If users do not specify the unit parameter on their DD statements or the dynamic allocation equivalent, SMS applies the default unit if the data set is non-system managed and has a disposition of either MOD (treated as NEW) or NEW. If you specify a default unit in the base configuration, make certain that it exists on the system performing the allocations. If you do not specify a default unit, end users must code the UNIT
parameter to allocate data sets that are not system managed.

When allocating space for a new data set on DASD, SMS converts all space requests in tracks (TRK) or cylinders (CYL) into requests for space in KB or MB. If a generic device type such as the 3380 is specified, SMS uses the device geometry for that generic device to convert tracks or cylinders into KB or MB. If an esoteric device type such as SYSDA or no UNIT is specified, SMS uses the default

device geometry to convert tracks and cylinders into KB or MB. If the users in your installations specify space in tracks or cylinder units, and they specify an esoteric UNIT or no UNIT, you must specify a default device geometry prior to converting these allocations to system-managed data sets.

After SMS converts space requests to KB or MB, the space values are passed to the ACS routines. The values are later used to determine the number of tracks or cylinders to allocate for the data set. The default device geometry does not apply to objects or data sets allocated on tape.

There is only one default device geometry for the entire SMS complex. Default device geometry is an installation's definition of how much space is represented by a TRK or a CYL when an esoteric unit or no unit is specified. The device geometry is the track size and number of tracks per cylinder for the device.

The device geometry for a 3380 is 47476 bytes/track, 15 tracks/cylinder. The device geometry for a 3390 is 56664 bytes/track, 15 tracks/cylinder. It is up to each installation to decide what values to use.

Bytes/Track represents the number of bytes per track that SMS uses on allocations. Tracks/Cylinder represents the number of tracks per cylinder that SMS uses on allocations.

In order to activate an SMS configuration on a system, that system's name or the name of the sysplex to which the system belongs must be specified in the base configuration information.

If you were using VSAM RLS, you would also define cache sets in the base configuration information. Cache sets are a way of grouping coupling facility cache structures such that those groups can then be specified in a storage class.

What is a Storage Group?

Storage groups represent the physical storage managed by SMS. This storage can be collections of DASD volumes, volumes in tape libraries, volumes in optical libraries, or virtual input/output (VIO) storage. A storage group, used with storage classes, separates the logical requirements of accessing data from the physical requirements to store the data. You can use storage group attributes to specify how the system should manage the storage group. You use the storage group ACS routine to assign a new data set or object to a storage group. You can assign multiple candidate storage groups (except for objects), in which case the system chooses a specific storage group from your list. Storage group definitions are not apparent to users. Only you, as
storage administrator, can define, alter, or display storage group definitions.

Storage groups apply only to SMS-managed data sets and objects. They represent the physical storage on which the data sets and objects reside. The physical groupings of devices, or pools, can be modified dynamically without the need for an IPL. There are six types of storage groups:

Pool: Contains SMS-managed DASD volumes and allows those volumes to be managed as a single entity. IBM recommends that all of the devices in a single storage group have the same device geometry.

There are two special types of pool storage groups that should be mentioned separately. The first is an overflow storage group. An overflow storage group is one that is designated handle periods of high demand for initial primary space allocations. Overflow storage groups are utilized when the enabled (non-overflow) storage groups are above high threshold.

The second is an extend storage group. An extend storage group is a pool storage group to which data sets from the primary storage group can be extended when there is an insufficient amount of storage on the primary storage group. A primary storage group is the storage group in which the initial allocation resides.

A storage group can be both an overflow and an extend storage group

Dummy: Contains volume serials of volumes no longer connected to the system which are treated as SMS-managed; allows existing JCL to function unchanged

VIO: Contains no volumes; allocates data sets to paging storage which simulates the activity of a DASD volume

Object: Contains optical volumes and DASD volumes used for objects

Object Backup: Contains optical volumes used for backup copies of objects

Tape: Contains SMS-managed private tape volumes.

All SMS-managed data sets and objects must have a storage group. Pool storage groups are of particular interest because they allow DASD volumes to be pooled and managed collectively. You can also define whether or not automatic migration, backup, and dump are allowed within the pool.

What is a Storage Class?

Storage classes apply only to SMS-managed data sets and objects. They allow you to define different levels of performance and availability services for your data sets. Using them, you can separate the level of service needed by a data set or object from its physical characteristics. Storage classes can supply such information as attributes for dynamic cache management, sequential data set striping, and concurrent copy.

It is the association of a storage class with a data set or object which causes the data set or object to be SMS-managed. Because of this, such functions as dynamic cache management and sequential data set

striping apply only to SMS-managed data sets. Data sets may be SMS-managed or non-SMS managed. Objects must be SMS-managed.

What is a Management Class?

Management classes apply only to SMS-managed DASD data sets and objects. They supply a list of data set migration, backup, and retention values for DASD data sets, as well as backup requirements and class transition criteria for objects. DFSMShsm uses the attributes defined in the management class for a data set to automatically provide both storage management and availability management. Management classes can supply such information as expiration attributes, migration attributes, and backup attributes.

Management classes are what allows system managed storage to provide space management and availability management at the data set level. An SMS-managed DASD data set may or may not have a management class. An object must have a management class.

What is a Data Class?

Data classes apply to both SMS-managed and non-SMS managed data sets. They do not apply to objects. They allow you to define allocation defaults and simplify allocations by using data class in place of many keywords. Data classes can supply such information as space parameters, DCB attributes, VSAM attributes, volume count, and tape attributes.

Any values you explicitly specify always override values specified in a data class. This is to prevent the system from modifying the intent of your allocation. In addition, since not all attributes apply to every data set organization, SMS uses only those data class attributes that have meaning for the data set being allocated.

What is an ACS Routine?

ACS routines can be used to determine the SMS classes and storage groups for data sets and objects in an SMS complex. For storage administrators, ACS routines automate and centralize the process of determining SMS classes and storage groups. ACS routines also help convert data sets to an SMS environment.

An object is assigned to a storage group when it is stored and remains in that storage group throughout its lifetime.

Chapter 15, "Writing ACS Routines" of the DFSMSdfp Storage Administration Reference (SC126-7331) lists the rules for programming in the ACS language.

You can use the DFSMS NaviQuest tool to help you design and test your ACS routines. First, you can create test cases to perform extensive testing against test data representing actual data sets. Then you can test ACS routines in batch, freeing the workstation for other work. See DFSMS/MVS NaviQuest User's Guide for further information.

Through ISMF, you can create and maintain as many as four ACS routines in an SCDS, one for each type of SMS class and one for storage groups. After you have activated an SMS configuration, SMS executes ACS routines for the following operations:

- JCL DD statements (DISP=NEW, DISP=MOD)
- Dynamic allocation requests (DISP=NEW, DISP=MOD for a nonexistent data set)
- DFSMSdss COPY, RESTORE, and CONVERTV commands
- DFSMShsm RECALL and RECOVER
- Access method services ALLOCATE, DEFINE, and IMPORT commands
- OAM processing for STORE, CHANGE, and class transition
- Local data set creation by remote application through Distributed FileManager/MVS
- MVS data sets or OS/390 UNIX System Service (OS/390 UNIX) files created by remote application through the OS/390 Network File System server

You write ACS routines using the ACS programming language, a high-level programming language. The language follows a procedural flow of implementation that consists mainly of filtering criteria, IF/THEN statements, and SELECT/WHEN statements. Using these relational statements, ACS routines determine SMS classes and storage groups according to allocation parameters, data set sizes, object or data set names, and other variables.

Note: For system-managed data sets, the storage group is required because there is no way to explicitly specify storage groups. The other routines are optional. For objects, the storage group, storage class and management class ACS routines are required. For tape, the storage group, storage class and data class ACS routines are required.

What Happens to an Allocation if I Create an SMS Configuration?

In a non-SMS environment, the UNIT parameter is required for nearly all allocations (an exception is VSAM allocations done using IDCAMS). Unless an esoteric is used consistently, this can result in a large number of device dependent allocations. With SMS and a minimal configuration, the UNIT parameter may be removed from all allocations. If you choose not to make JCL changes, UNIT is ignored for SMS allocations, although it still applies to any non-SMS allocations.

The UNIT parameter is never required for SMS-managed allocations. This is because SMS selects the actual volume(s) for the allocation, and in doing so also selects a unit type. The UNIT parameter is required for non-SMS allocations. However, it may be taken from the default unit in the SMS base configuration information rather than from explicit user specification. This has two benefits:

1. It allows most allocations to be done independent of specific unit type (an exception might be an absolute track allocation).

2. Users need not know ahead of time whether or not allocations will be SMS-managed. With default unit, the unit may be omitted in either case.

In addition, the default device geometry in the SMS base configuration information may be used when allocating new data sets to convert CYL or TRK requests into KB or MB. You can use the default device geometry to specify the default number of bytes per track and tracks per cylinder. SMS converts all space requests in tracks (TRK) or cylinders (CYL) into requests for space in KB or MB.

If a generic device type such as the 3380 is specified, SMS uses the device geometry for that generic device to convert tracks or cylinders into KB or MB. If an esoteric device type such as SYSDA or no UNIT is specified, SMS uses the default device geometry to convert tracks and cylinders into KB or MB. If the users specify space in tracks or cylinders, and they specify an esoteric UNIT or no UNIT, a default device geometry must be provided prior to converting these allocations to system-managed data sets.

The advantage to using the default device geometry is that it allows a consistent amount of space to be allocated regardless of device type when an allocation is done in device-dependent units. However, some care must be taken when using it. If the default device geometry you specify does not match the geometry for which the allocation was originally intended, it may result in either more or less space being allocated that was actually required.

System Overview

The SMS code resides in the extended link pack area.

SMS adds information about a data set in the same manner as Catalog or RACF.
a. This information is not part of the data set
b. It describes the data set and the DASD, optical disk, or tape where it is stored.

SMS information is stored on DASD in the Catalog, RACF data set, or in the SMS configuration.

SMS may be entered either by BALR or using the MVS subsystem interface.

Defining the Control Data Sets

SMS control data sets can be either SMS-managed or non-SMS-managed. Initially you should ensure that your control data sets have a volume count of one. The volume count can be either explicitly specified, implied by the number of volume serials provided, or derived from the data class assigned to the data set. If you give an SMS control data set a volume count that is greater than the number of volumes on which it actually resides you might receive messages IEF244I and IEF489I when you attempt to activate it.

If you have a multivolume SCDS which you are activating into a single volume ACDS, you might receive an error because the ACDS is not large enough and volumes cannot be dynamically added to it. To bypass this problem, you need to create a new multivolume ACDS and then activate the ACDS and SCDS simultaneously using the SETSMS command.

If your SMS complex includes more than 16 systems, be sure that the ACDS and COMMDS are accessible to every system in the complex. Define your control data sets on volumes which are capable of being attached to more than 16 systems, such as IBM RAMAC Virtual Array volumes.

The example shown allocates a 6-track VSAM linear SCDS (source control data set) named SMS.SCDS1.SCDS, a 6-track VSAM linear ACDS (active control data set) named SMS.ACDS1.ACDS, and a 1-track VSAM linear COMMDS (communications data set). After allocating an SCDS, you define its contents through ISMF dialogs.

You should allocate the SMS control data sets on devices shared by all systems in the SMS complex. If you allocate an SCDS on a device that is not shared by all the systems, then you can activate the SCDS only from systems that have access to it.

You should specify the REUSE option when you define an SCDS or ACDS to avoid running into space problems (SMS reason code 6068) as result of subsequent SCDS updates, or IMPORT/EXPORT functions

ACDSs and COMMDSs must reside on a shared volume, accessible from all systems in the SMS complex. To ease recovery in case of failure, the ACDS should reside on a different volume than the COMMDS. Also, you should allocate a spare ACDS and COMMDS on a different shared volume.

You create the contents of an ACDS by activating a valid SCDS. The distinction between a valid SCDS and an invalid one is described in z/OS DFSMSdfp Storage Administration Reference (SC26-7402). Information on sizing your control data sets can also be found in this book.

The control data sets (ACDS and COMMDS) must reside on a volume that is not reserved by other systems for a long period of time because they must be available to access for SMS processing to continue.

We recommend that you use SHAREOPTIONS(3,3) when allocating an ACDS or COMMDS. This allows full authority to read from and write to an ACDS from any system. The ACDS and COMMDS must be accessed from all systems in the complex simultaneously.

SYS1.PARMLIB: IGDSMSxx

For each system in the SMS complex, you must create an IGDSMSxx member in SYS1.PARMLIB. The IGDSMSxx member contains SMS initialization control information, where xx is any value allowed by the naming conventions for SYS1.PARMLIB members. It has a default value of 00.

Every SMS system must have an IGDSMSxx member in SYS1.PARMLIB that specifies a required ACDS and COMMDS pair. This ACDS and COMMDS pair is used if the COMMDS of the pair does not point to another COMMDS.

If the COMMDS points to another COMMDS, the referenced COMMDS is used. This referenced COMMDS might contain the name of an ACDS that is different from the one specified in the SYS1.PARMLIB member. If so, the name of the ACDS is obtained from the COMMDS rather than from the IGDSMSxx member in SYS1.PARMLIB to ensure that the system is always running under the most recent ACDS and COMMDS.

If the COMMDS of the ACDS and COMMDS pair refers to another COMMDS during IPL, it means a more recent COMMDS has been used. SMS uses the most recent COMMDS to ensure that you cannot IPL with a down-level configuration.

The data sets that you specify for the ACDS and COMMDS pair must be the same for every system in an SMS complex. Whenever you change the ACDS or COMMDS, update the IGDSMSxx member of SYS1.PARMLIB for every system in the SMS complex so that it specifies the same data sets.

The IGDSMSxx member contains all the parameters shown. These parameters direct the SMS initialization process and specify the names of the ACDS and COMMDS.

ACDS, COMMDS, REVERIFY, and ACSDEFAULTS need to be the same for all your systems. The rest of the parameters can be different. For more information on the parameters of the IGDSMSxx member of SYS1.PARMLIB, see z/OS MVS
Initialization and Tuning Guide.

The only required keywords are SMS, ACDS, and COMMDS. All the rest are optional and most will default if not specified.

SMS Identifies the member as a repository of SMS initialization control information.

ACDS(dsname) identifies the name of the data set containing the active configuration. If you omit dsname, the operator is prompted for a value.

COMMDS(dsname) identifies the name of the COMMDS. If you omit dsname, the operator is prompted for a value.

ACSDEFAULTS({YES|NO}) indicates whether SMS initializes the following ACS routine variables from an additional call to the Resource Access Control Facility (RACF), a component of the SecureWay Security Server for z/OS:

&APPLIC
&DEF_DATACLAS
&DEF_MGMTCLAS
&DEF_STORCLAS

If you specify NO, these variables have no values associated with them. The default value for ACSDEFAULTS is NO. The ACSDEFAULTS keyword is not applicable for OAM.

Note: The ACSDEFAULTS keyword is not applicable when USE_RESOWNER(NO) keyword is also specified.

AKP specifies the activity keypoint trigger value, which is the number of logging operations between keypoints. Up to 32 activity keypoint values can be specified. AKP values must be specified in the same order as DFSMStvs instance names. Specify a value from 200 to 65535.

You can allow some values to default while specifying others. For example,

AKP(800,,3000)

In this example, the value for the DFSMStvs instance in the second position defaults to 1000, and sets the values for the first and third DFSMStvs instances to 800 and 3000 respectively.
If AKP is specified with only the TVSNAME parameter without an identifier, AKP applies to the system on which the PARMLIB member is being read. TVSNAME must be specified with the AKP parameter.

Activity keypointing is the process of accounting for all the log records in the undo and shunt logs that are involved in active units of recovery. Activity keypointing enables DFSMStvs to delete records that are no longer involved in active units of recovery. It also enables DFSMStvs to update the undo log to optimize its restart performance related to reading the log stream. The default is 1000.

ASID({asid|*}) limits tracing to a certain address space or permits it for all address spaces. Specify * if you want all address spaces traced (providing SMS tracing is activate). This is the default. You can enter up to 4 digits for the ASID keyword. If you leave off the leading zeroes, they are inserted.

BMFTIME({nnn|3600}) specifies the number of seconds between recording SMF record type 42, subtype 1 records for PDSE (where the buffer hits are I/O requests which did not require actual disk I/O). You can specify a value from 1-86399 (23 hours, 59 minutes, 59 seconds). The default is 3600 (1 hour).

CACHETIME({nnn|3600}) specifies the number of seconds between recording SMF records for device cache use. The CACHETIME parameter applies only to the volumes behind an IBM 3990 Storage Control with cache unit. You can specify a value from 1-86399 (23 hours, 59 minutes, 59 seconds). The default is 3600 (1 hour).

CF_TIME({nnn|3600}) indicates the number of seconds between recording SMF records for the coupling facility (both cache and lock). You can specify a value from 1-86399 (23 hours, 59 minutes, 59 seconds). The default is 3600 (1 hour). This keyword sets the interval time for the following SMF 42 subtypes:

SUBTYPE 15 CF storage class average response time
SUBTYPE 16 CF data set average response time
SUBTYPE 17 CF lock structure activity
SUBTYPE 18 CF cache partition summary
SUBTYPE 19 SMSVSAM LRU statistics summary

COMPRESS({TAILORED|GENERIC}) specifies the type of compression to be used for the data set.

TAILORED specifies that the data set is eligible for compression specifically tailored to the data set. A tailored dictionary is built, using the initial data written to the data set, and imbedded into the data set. The dictionary is used to compress or expand data written to or read from the data set. This type of compression applies only to sequential data sets, not to VSAM KSDSs.

To convert an existing DBB-based compressed data set to use tailored compression, you must set the COMPRESS parameter to TAILORED and copy the generic DBB-based data set to a new data set that meets compression requirements. You can use IEBGENER, ICEGENER, REPRO, or any QSAM or BSAM application to copy the DBB-based data set to a new data set that meets compression requirements.

Use tailored compression only when all systems in the SMS complex have been converted to DFSMS/MVS V1R4 or higher, and when there is no need to revert to a prior release level for local recovery or remote recovery with Aggregate Backup and Recovery Support (ABARS).

GENERIC specifies that the data set be compressed using

generic Dictionary Building Block (DBB) compression. The dictionary is derived from a defined set of compression algorithms in data set SYS1.DBBLIB. This is the default.

DB2SSID(ssid) identifies either the name of the DB2 subsystem used by OAM for object storage, or a group attachment name if data sharing is being used. If you specify a group attachment name, the DB2 startup ECB is ignored by DB2, so DB2 must be initialized before OAM is started.

If OAM object support and DB2 are not needed for OAM's object directories, object data tables, and configuration database, this parameter is not necessary and should not be specified. The ssid can be from one to four characters and there is no default.

DEADLOCK_DETECTION({iiii|15,kkkk|4}) specifies the deadlock detection intervals used by the DFSMSdfp Storage Management Locking Services. The first subparameter is the local deadlock detection cycle and specifies the interval in seconds for detecting deadlocks within a system. The second
subparameter is the global deadlock detection cycle and specifies the interval for detecting deadlocks between systems. The value is specified as the number of local detection cycles that occur before global deadlock detection is initiated.

iiii one to four digit numeric value in the range 1-9999 that specifies the length in seconds of the local deadlock detection interval. The default is 15 seconds.

kkkk one to four digit numeric value in the range 1-9999 that specifies the number of local deadlock cycles that must complete before global deadlock detection occurs. The default is 4.

DESELECT({event[,event][,...]|ALL}) deletes items from the list of events to be traced (if SMS tracing is active). DESELECT has no default. If you specify events that conflict in SELECT and DESELECT, the keyword that appears last has final authority. The events that you can specify on SELECT and DESELECT are:

MODULE SMS module entry or exit
DSTACK SMS Data Set Stacking Service
SMSSJF SMS/SJF interfaces
SMSSSI SMS/SSI interfaces
ACSINT ACS services interfaces
OPCMD Operator commands
CONFC Configuration change
CDSC Control data set changes

CONFS SMS Configuration services
MSG SMS Message services
ERR SMS Error recovery and recording services
CONFR Return data from an active configuration
CONFA Activate a new configuration
ACSPRO Perform ACS processing
IDAX SMS interpreter/dynamic allocation
DISP SMS disposition processing exit
CATG SMS catalog services
VOLREF SMS VOLREF services
SCHEDP SMS scheduling services (preallocate catalog orientation)
SCHEDS SMS scheduling services (system select)
VTOCL SMS VTOC/data set services (allocate existing data set)
VTOCD SMS VTOC/data set services (delete existing data set)
VTOCR SMS VTOC/data set services (rename existing data set)
VTOCC SMS VTOC/data set services (allocate new data set)
VTOCA SMS VTOC/data set services (add a volume to a data set)
RCD SMS Recording services
DCF SMS device control facility
DPN SMS device pool name select subsystem interface
TVR SMS tape volume record update facility
ALL All of the above options

DINTERVAL({nnn|150}) specifies the number of seconds SMS allows to elapse before it reads device statistics. The DINTERVAL parameter applies only to the volumes behind an IBM 3990 Storage Control with cache unit. You can specify a value from 1-999 (16 minutes, 39 seconds). The default is 150. SMS uses device statistics to manage hardware usage and maximize efficiency.

DSNTYPE({LIBRARY|PDS}) specifies the installation default for data sets allocated with directory space but without a data set type specified. If the data set type is PDS, the default is a partitioned data set format; if the data set type is LIBRARY, the default is a PDSE (partitioned data set extended) format.

DSSTIMEOUT specifies the number of seconds that the dss component of DFMSMS will wait during backup processing for quiesce data set requests to complete. Specify a value from zero to 65536

seconds (which is more than 18 hours). If you specify a value between 1 and 299 seconds, the system uses a value of 300 seconds (which equals 5 minutes).

The value specified in the DSSTIMEOUT parameter value is activated when the first instance of the SMSVSAM address becomes active in the sysplex. All subsequent SMSVSAM instances will use the same value. The default is 0.

You can alter the DSSTIMEOUT value dynamically in the following ways:

- Using the SETSMS DSSTIMEOUT(nnnn) command
- By adding or updating the DSSTIMEOUT parameter in IGDSMSxx parmlib member and then activating it with the SET SMS=xx command

GDS_RECLAIM specifies whether the system will reclaim generation data sets. This is an optional keyword.

If you specify YES, the system will reclaim generation data sets.

If you specify NO, GDS reclaim processing will not be done. This means that if a new generation is created by a job but does not get rolled in, then you must manually:

- delete the generation,
- rename it to a different name, or
- roll it in using Access Method Services

If you do not take one of these manual steps, then on a subsequent attempt to create a new generation, SMS will detect the existence of a duplicate data set and will fail the allocation. The default is yes.

HSP_SIZE(nnn) specifies the size, in megabytes, of the hiperspace that is used to cache PDSE member pages. You can specify a hiperspace size up to 512 MB. The valid range for HSP_SIZE is 0 to 512. If you specify 0, the hiperspace is not created. To create a hiperspace, your system must have expanded storage available. If expanded storage is available and you do not specify a value for HSP_SIZE, the default is 256 MB or half of expanded storage, whichever is less.
INTERVAL({nnn|15}) specifies the synchronization time interval of the system, which is the number of seconds between system checks of the COMMDS for news about SMS configuration changes from other systems in the SMS complex. You can specify a value from 1-999 (16 minutes, 39 seconds). The default is 15.

How soon OAM is notified of an SCDS activation, and therefore the delay until OAM will restart, depends on the time interval specified with this INTERVAL keyword. This may apply to other address spaces that are also dependent on the SMS configuration.
JOBNAME({jobname|*}) limits tracing to a certain job or permits tracing on all jobs. The default is to trace all jobs, *.

LOG_OF_LOGS specifies the log stream that is to be used as the log of logs. This log contains copies of tie-up and file-close records written to forward recovery logs and is used by forward recovery products. If this parameter is not specified, no log of logs is used. This parameter is unique to each system in the sysplex. As each instance of DFSMStvs starts, it uses the log of logs name that is found in its member of SYS1.PARMLIB. If you run test and production systems within the same sysplex, use a separate PARMLIB member for the test system and specify a different log of logs in it.

If you specify the LOG_OF_LOGS parameter, you must also specify the TVSNAME(nnn) parameter. There is no default for this parameter.

LRUCYCLES(nnn|240) specifies the maximum number of times (5 to 240 cycles) that the buffer management facility (BMF) least recently used (LRU) routine will pass over inactive buffers before making them available for reuse. While this parameter sets the maximum value, BMF will dynamically change the actual number of times it passes over inactive buffers.

LRUCYCLES is related to LRUTIME. A change to the LRUCYCLES value introduced by this parameter will take effect on the next execution of the LRU routine. The default is 240. Most installations use the default.

LRUTIME(nnn|15) specifies the number of seconds (5 to 60) that the buffer management facility (BMF) will wait between calls to the BMF data space cache LRU (least recently used) routine. The LRU routine releases inactive buffers in the BMF data space that are used to cache PDSE (partitioned data set extended) directory data. LRUTIME is related to LRUCYCLES. A change to the LRUTIME value introduced by this parameter will take effect on the next execution of the LRU routine. The default is 15.

MAXLOCKS(max|0,incr|0) specifies the maximum number of locks a single transaction may hold before a warning message is issued. Once the maximum number of unique lock requests is reached, additional warning messages will be issued every time the number of unique lock requests over and above the maximum increases by a multiple of the increment value.

OAMPROC(procname) identifies the procedure name that starts the OAM address space when SMS is initialized. You must specify this keyword if you want the OAM address space started during IPL. The procedure name can be from one to eight characters and there is no default.

OAMTASK(taskname) specifies the task ID that is used to start the OAM address space. OAMTASK is optional; if it is specified without OAMPROC it is ignored. If omitted, the task ID defaults to the procedure name specified in OAMPROC. OAM keywords take effect only if you initialize SMS at IPL; otherwise they are not saved. The task ID can be from one to eight characters.

OVRD_EXPDT({YES|NO}) allows for the override of an expiration date when an unexpired SMS-managed DASD data set is deleted. If you specify NO, expiration dates are honored unless specific action is taken to override them. If you specify YES, any expiration date specified for the data set is overridden when deletion of an SMS-managed DASD data set is attempted through JCL, SVC 99, IEHPROGM or the SCRATCH macro. YES should only be used when management class cannot be used to override expiration dates. It is intended to help with data sets which are converted from tape to DASD since expiration date is more commonly used with tape.

PDSESHARING({NORMAL|EXTENDED}) specifies whether PDSEs can be shared by all systems within a Parallel Sysplex for input only or for input and output. A value of NORMAL means that PDSEs can be shared by all the systems in a Parallel Sysplex but only for input. A value of EXTENDED means that PDSEs can be shared by all the systems in a Parallel Sysplex for input and output. The default is NORMAL.

PDSE_MONITOR({YES|NO}[,interval[,duration]]) specifies how the processing for the PDSE monitor is started or modified. To turn monitor processing on, specify YES. To turn monitor processing off, specify NO. The default is YES. For interval, specify the number of seconds between successive scans of the monitor. If you do not specify a value, the interval is unchanged except at IPL, when it is set to 60. For duration, specify the number of seconds a possible error condition must exist before it is treated as an error. If you do not specify a value, the duration is unchanged except at IPL, when it is set to 15.

QTIMEOUT specifies the quiesce exit timeout value, in seconds. The quiesce timeout value specifies the amount of time the DFSMStvs quiesce exits allow to elapse before concluding that a quiesce cannot be completed successfully. Only one quiesce timeout value can be specified. The first instance of DFSMStvs that is brought up within the sysplex determines the value. Subsequent SFSMStvs instances use the value established by the first system, regardless of what can be specified in their members of SYS1.PARMLIB. Specify a value between 60 and 3600.

If you specify the QTIMEOUT parameter, you must also specify the TVSNAME(nnn) parameter. This parameter applies across all systems. The default is 300 seconds.

REVERIFY({YES|NO}) specifies whether SMS verifies authorization only at job interpretation time or at both job interpretation and job execution time. A value of NO instructs SMS to verify a user's authority to allocate a new data set, to use a storage class, and to use a management class at job interpretation time only. A value of YES instructs SMS to perform the verification at both job interpretation and job execution time. JES2 environments ignore the REVERIFY keyword. It has a default value of NO.

[RLSINIT({NO|YES})] specify YES if you want the SMSVSAM address space started as part of system initialization or the V SMS,SMSVSAM,ACTIVE command. This value applies only to the system accessed by the parmlib member and is acted upon when SMSVSAM is next started. The default is NO.

RLS_MAXCfFeatureLevel({A|Z}) specifies the method that VSAM RLS uses to determine the size of the data that is placed in the CF cache structure. If you specify A, caching proceeds using the RLS CF Cache Value keyword characteristics that are specified in the SMS data class that is defined for the VSAM sphere. If you do not specify a value, or if you specify Z, then only VSAM RLS data that have a Control Interval (CI) value of 4K or less are placed in the CF cache structure. The default is Z.

If A is specified for the RLS_MaxCfFeatureLevel parameter, systems lower than V1R3 will not be able to connect to the CF cache structure. If a lower-level system is the first system activated in the sysplex, RLS_MaxCfFeatureLevel defaults to Z, and all systems will be able to connect to the CF cache structure. If the SETSMS command is used to change the RLS_MaxCfFeatureLevel value to A on a mixed-level system, the command is rejected and message IGW500I is issued.

RLS_MAX_POOL_SIZE({nnnn|100}) specifies the maximum size in megabytes of the SMSVSAM local buffer pool. SMSVSAM attempts to not exceed the buffer pool size you specify, although more storage might be temporarily used. Because SMSVSAM manages buffer pool space dynamically, this value does not set a static size for the buffer pool.

Use SMF 42, subtype 19 records to help you determine the maximum size of the SMSVSAM local buffer pool.

You can specify a two to four-digit numeric value, with 10 as
the minimum value. If you specify a value less than 10, the field is set to 10. If you specify a value greater than 1500, SMSVSAM assumes there is no maximum limit. We recommend that you limit the size of the local buffer pool. The default is 100 MB.

RLSTMOUT({nnn|0}) specifies the maximum time, in seconds, that a VSAM RLS request must wait for a required lock before the request is assumed to be in deadlock. You can specify a value between 0 to 9999 (in seconds). A value of 0 means that the VSAM RLS request has no time out value and the request will wait for as long as necessary to obtain the required lock. RLSTMOUT can be specified only once in a sysplex and applies across all systems in the sysplex. The default is 300.

SELECT({event[, event][,...]|ALL}) adds items to the list of events to be traced (if SMS tracing is active). The default is ALL. See DESELECT for a list of valid events.

SIZE(nnn{K|M}) specifies the size of the SMS trace table in bytes. When the unit is K, the value can range from 0K to 255000K, and it is rounded up to the nearest 4K unit. When the unit is M, the value can range from 0M - 255M. If you specify a value of 0, no tracing is performed. The default is 128K.

SMF_TIME({YES|NO}) YES indicates that the following SMF type 42 records are created at the SMF interval time, and that all of the indicated records are synchronized with SMF and RMF(TM) data intervals:

SUBTYPE 1 Buffer management statistics
SUBTYPE 2 Cache control unit statistics (IBM 3990-3, 3990-6, IBM RVA, and IBM ESS)
SUBTYPE 15 CF storage class average response time
SUBTYPE 16 CF data set average response time
SUBTYPE 17 CF lock structure activity
SUBTYPE 18 CF cache partition summary
SUBTYPE 19 SMSVSAM LRU statistics summary

DFSMS creates the specified SMF record at the end of the interval period and SMF sends the event notification signal. If YES is specified, this subparameter overrides the following subparameters: BMFTIME, CACHETIME, CF_TIME. YES is the default. See z/OS MVS System Management Facilities (SMF) for more information on SMF.

SYSNAME specifies the name of the system or systems on which DFSMStvs instances are to run. You can specify up to 32 system names, which must be in the same order as the DFSMStvs instance names. SMS examines the specified system names and compares them to the system names in the CVT. When it finds a match, SMS stores the value of the TVSNAME parameter in the matching position as the DFSMStvs instance name for the system.
Use the combination of SYSNAME and TVSNAME when the PARMLIB member is shared between systems. If no SYSNAME parameter is specified, the TVSNAME parameter applies to the system on which the PARMLIB member is read.

If you specify the SYSNAME parameter, you must also specify the TVSNAME(nnn) parameter. If you specify SYSNAME, but the system name is not specified, DFSMStvs is not started on the system. There is no default for this parameter.

SYSTEMS({32|8}) indicates whether the system is running in compatibility mode (eight name limit) or 32-name mode. The default is 8.

SYSTEMS(8) specifies that a maximum of eight system names, system group names, or both, can be specified in the SMS configuration. This indicates that the system is running in compatibility mode and can share configurations (SCDSs or ACDSs) and COMMDSs with systems that are running down-level releases of DFSMS or DFP. Essentially, the system continues to operate as it has in the past.

SYSTEMS(32) specifies that a maximum of 32 system names, system group names or both can be specified in the SMS configuration. This indicates the system is not running in compatibility mode.

Note: Therefore the ACDS, SCDS and COMMDS must not be shared with the systems that are running down-level releases of DFSMS or DFP or that are running in compatibility mode.

TRACE({OFF|ON}) indicates whether the SMS trace facility is activated. The default is ON. The SMS trace facility records trace records in the SMS address space. See z/OS DFSMSdfp Diagnosis Reference for additional information.

TRACEEXIT(user_trace_exit) specifies the name of your trace exit routine. You can specify an alphanumeric value of 1 - 8 characters, starting with an alphabetic character. Your value must represent a valid module name. The default value is blank. This interface is usually used only for diagnostic purposes under the guidance of the service organization.

TVSNAME specifies the identifier or identifiers that uniquely identify DFSMStvs instances that run in the sysplex. You can specify up to 32 identifiers, each of which must be a numeric value from 0 to 255. Each identifier must be unique within the sysplex. DFSMStvs uses this identifier as the last byte of its instance name (although it displays as three bytes). If the TVSNAME parameter is specified without the SYSNAME parameter, only a single value can be specified, and the name applies to the system on which the PARMLIB member is read. The numeric digits are appended to the characters IGWTV to form the DFSMStvs instance name.
You can use the TVSNAME parameter without the SYSNAME parameter when the PARMLIB member is not shared between systems. If no TVSNAME parameter is found, DFSMStvs processing is not available on the system. There is no default for this parameter.
TV_START_TYPE specifies the type of start that each instance of DFSMStvs is to perform. Up to 32 TV_START_TYPE values can be specified. TV_START_TYPE values must be specified in the same order as DFSMStvs instance names. If WARM is specified, DFSMStvs reads its undo log and processes the information found in accordance with the information that RRS has about any outstanding units of recovery. If COLD is specified, DFSMStvs deletes any information remaining in the undo log and starts as if the log were empty. Use COLD only when the DFSMStvs undo log has been damaged.

Recommendation: Do not cold start DFSMStvs unless the DFSMStvs system logs have been damaged. After a cold start, data sets for which recovery was not completed could be left in a damaged state and must be recovered manually. If the data sets are forward recoverable, their forward recovery logs might also be damaged. IBM recommends that you manually recover the data sets (without using forward recovery), take back ups of them and any other data sets using the forward recovery log, then delete and redefine the forward recovery log.

You can allow some values to default and others to be specified. For example,

TV_START_TYPE(COLD,,COLD)

Specifying the keyword this way would cause the DFSMStvs instance in the second position to warm start, while the first and third DFSMStvs instances would cold start.

If both TVSNAME and SYSNAME parameters are also specified, TV_START_TYPE applies to the system specified on the SYSNAME parameter preceding or following it. If TV_START_TYPE is

found with only the TVSNAME parameter, it applies to only the system on which the PARMLIB member is being read. The default is WARM.

TYPE({ALL|ERROR}) specifies whether SMS traces all events or only errors. The default is ERROR.

USE_RESOWNER({YES|NO}) indicates whether construct authorization checking is done using the RESOWNER value, which is based on the high-level qualifier of the data set name, or using the data set allocator user ID. If you specify NO, the RESOWNER value is not extracted and the allocator user ID is used. The default value for USE_RESOWNER is YES.

SYS1.PARMLIB: IEFSSNyy

Before you can activate SMS, MVS must recognize SMS as a valid subsystem. You can define SMS to MVS with IEFSSNxx.

You can define SMS to MVS through IEFSSNxx in one of two ways. The first method involves placing a record for SMS in the existing IEFSSNxx member that is used for the next IPL.

The second method involves creating a new IEFSSNxx member that contains only the record for SMS. To identify the IEFSSNxx member containing the SMS entry, you must update the IEASYSyy member that you use to IPL. In the SSN parameter of IEASYSyy, the suffix identifying the IEFSSNxx member that contains the SMS entry should appear first, because SMS must be active before starting any other subsystem. For example, in:

```
SSN=(00,01,...)
```

the 00 identifies IEFSSN00 as the first member, and 01 identifies IEFSSN01 as the second member, and so forth. You can specify additional members to define more subsystems. For example:

```
SSN=(00,01,02,03).
```

You have the option of manually starting SMS with a command or automatically starting it at future IPLs. When you are first preparing for SMS, you might want to omit the IGDSSIIN module name from the SMS entry. This defines SMS to MVS at the next IPL but does not automatically start SMS. After you IPL the system, SMS is defined as a subsystem and you can then start it manually.

To define SMS to MVS, you must place a record for SMS in an IEFSSNxx member. You can code an IEFSSNxx member with keyword or positional parameters, but not both. We recommend using

keyword parameters. Note also that you must use the same format for all systems on which you are running SMS.

The second example shows the use positional instead of keyword parameters.

The SMS keyword is required and identifies and defines the Storage Management Subsystem to MVS. The other keywords are optional.

IGDSSIIN identifies the subsystem initialization routine IGDSSIIN for SMS. If you include it in the SMS entry, SMS is automatically started during every future IPL. If you omit it from the SMS entry, SMS is defined as a valid subsystem to MVS, but does not automatically start during future IPLs.

ID={xx|00} specifies the SYS1.PARMLIB member used for initialization in two special cases:

1. When the value specified in the SMS parameter of IEASYSyy does not correspond to any IGDSMSxx and the default IGDSMS00 does not exist.

2. When initialization of software controlling PDSEs fails.

To use PDSEs, SMS and the PDSE address space must be active. SMS allows you to allocate PDSEs and the PDSE address space allows you to open, read, write, and update PDSEs. If the PDSE address space initializes, the IGDSMSxx member used for SMS initialization is determined by the SMS=xx parameter in IEASYSyy. If the PDSE address space fails, the IGDSMSxx member used for SMS initialization is determined by the ID=xx parameter in IEFSSNxx. The SMS parameters should be the same unless you want different parameters for SMS when PDSEs are not available.

If you specify both ID and PROMPT, enclose them in single quotation marks and separate them with a comma.

PROMPT={DISPLAY|YES|NO} indicates how much control you want the operator to have during the rest of SMS initialization. DISPLAY displays the contents of the IGDSMSxx member, but the operator cannot modify them. YES displays the contents of the IGDSMSxx member and prompts the operator to modify the parameters in the IGDSMSxx member. The modifications take effect for the current IPL, but the contents of the IGDSMSxx SYS1.PARMLIB member do not change. NO, which is the default, neither displays the parameters nor allows the operator to modify them.

If you specify more than one keyword, enclose them in single quotation marks and separate them with a comma.

SYS1.PARMLIB: IEASYSzz

Normally, it is the SMS member specified here rather than the one specified in the IEFSSNyy member which controls SMS initialization. You should specify the same value in SMS=xx in IEASYSyy and in ID=xx in IEFSSNxx to avoid confusion.

Setting Storage Administrator Options

The first time you select ISMF, you get the ISMF Primary Option Menu for end users. To get the ISMF Primary Option Menu for storage administrators, select option 0, ISMF PROFILE. Within the ISMF Profile Option Menu, select option 0, USER MODE, and press ENTER. You get the User Mode Entry panel, where you indicate that you want the storage administrator Primary Option Menu for all future ISMF sessions. To do this, select option 2 on the User Mode Entry panel. After changing the user mode, you must exit ISMF and then return to it to view the Primary Option Menu for Storage Administrators.

You will probably want to prevent end users from gaining access to the storage administrator primary option menu. Use can use RACF to do this. The procedure for doing so is described in chapter 13 of the DFSMSdfp Storage Administration Reference.

Defining the Base Configuration

The first thing that you define in an SCDS is the base configuration. A base configuration contains installation defaults, such as a default management class, and identifies the systems to which the SMS configuration applies.

You need to determine the system names and system group names that you want to specify in the SCDS. These systems and system groups constitute the SMS complex.

Before defining the base configuration, consider what you want to do with system-managed data sets that do not have a management class. You can specify a default management class for these data sets in the base configuration. You can create the management class any time before validating the SCDS.

If the management class ACS routine does not determine a management class for a data set, DFSMShsm processes the data set using the default management class, if one exists. If you do not specify a default management class in your base configuration, DFSMShsm uses its own defaults.

If the management class ACS routine does not determine a management class for an object, the OSREQ request fails. This request can be OSREQ STORE, CHANGE, or OSMC class transition processing.

After allocating an SCDS, you can define a base configuration. By selecting option 8 from the ISMF Primary Option Menu for Storage Administrators, you can invoke the Control Data Set (CDS) Application Selection panel.

CDS Application Selection

On this panel, you can specify the name of the SCDS that is to contain the base configuration. ISMF primes the CDS Name field with the last used SCDS name.

To define a base configuration, select option 2, Define, and press Enter.

Defining the Base

As its name implies, an SCDS base configuration forms the foundation for building an SCDS.

For system-managed data sets that have not been assigned a management class, DFSMShsm uses the default management class for expiration, migration, and backup information. The default management class does not apply to objects. You specify the name of the management class in the Default Management Class field.

The default management class name is not saved in the data set catalog entry, so that you can tell the difference between data sets with assigned management classes and those without them. Specifying a default management class is optional. If a data set has no management classes and a default management class is not defined, DFSMShsm uses its own defaults for the data set.

The default unit is an esoteric or generic device name, such as SYSDA or 3390, that applies to data sets that are not managed by SMS. For new data set allocations, the default unit allows users to omit the UNIT parameter from DD statements or dynamic allocation equivalent, as they can when allocating system-managed data sets. The default unit does not apply to objects.

If users do not specify the unit parameter on their DD statements or the dynamic allocation equivalent, SMS applies the default unit if the data set is non-system managed and has a disposition of either MOD (treated as NEW) or NEW. If you specify a default unit in the base configuration, make certain that it exists on the system performing the allocations. If you do not specify a default unit, end users must code the UNIT parameter to allocate data sets that are not system-managed.

When allocating space for a new data set on DASD, SMS converts all space requests in tracks (TRK) or cylinders (CYL) into requests for space in KB or MB. If a generic device type such as the 3380 is specified, SMS uses the device geometry for that generic device to convert tracks or cylinders into KB or MB. If an esoteric device type such as SYSDA or no UNIT is specified, SMS uses the default device geometry to convert tracks and cylinders into KB or MB. If the users in your installations specify

space in tracks or cylinder units, and they specify an esoteric UNIT or no UNIT, you must specify a default device geometry prior to converting these allocations to system-managed data sets.

After SMS converts space requests to KB or MB, the space values are passed to the ACS routines. The values are later used to determine the number of tracks or cylinders to allocate for the data set. The default device geometry does not apply to objects or data sets allocated on tape.
There is only one default device geometry for the entire SMS complex. Default device geometry is an installation's definition of how much space is represented by a TRK or a CYL when an esoteric unit or no unit is specified. The device geometry is the track size and number of tracks per cylinder for the device.

The device geometry for 3380 is 47476 bytes per track, 15 tracks per cylinder. The device geometry for 3390 is 56664 bytes per track, 15 tracks per cylinder. It is up to each installation to decide what values to use.

Defining SMS Classes and Groups

You will need to define at least one storage class (option 5, then option 3), at least one storage group with at least one volume (option 6, then option 3) and a storage group ACS routine (option 7, then option 1 to edit, option 2 to translate and option 3 to validate) in order for your configuration to be valid. Management classes, which are used to define space management and availability management policy, and data classes, which are used to simplify allocations, are optional. The storage class ACS routine, management class ACS routine, and data class ACS routine are likewise optional, since all of them can be specified on allocations and JCL. However, if you want to maintain control of what is SMS-managed and what is not, then it is recommended that you provide a storage class ACS routine to control what storage classes are assigned to what data sets. It is the association of a storage class with a data set which causes the data set to be SMS-managed.

Defining ACS Routines

Before you start writing ACS routines, you need to allocate data sets to contain the source. They can be stored as physical sequential data sets or as members in PDSs or PDSEs. It is frequently useful to use a PDS or PDSE so that it can serve as the container for all your ACS routine source.

The ACS routines are one of the mechanisms DFSMS/MVS provides to help automate storage management. Using ACS routines, you can centralize control of storage management and automatically associate SMS classes and groups with data sets.

An ACS routine is a sequence of instructions for having the system assign SMS classes and groups to data sets and objects. The selection of specific classes and groups is based on information from JCL or other allocation parameters. ACS routines can use parameters, such as data set name, volume serial number, job name, and data set size, to assign classes and groups to data sets.

There are four types of ACS routines. There can be one ACS routine of each type in an SMS configuration. ACS routines are executed whenever data sets and objects are created, either at initial allocation or due to an operation such as RECALL, RECOVER, COPY, or RESTORE. They are executed in the following order:

1. data class

The data class routine can assign data classes for both SMS-managed and non-SMS-managed data sets. This allows users to simplify allocations by using supplied values for JCL parameters or DCB attributes. The data class ACS routine is run only at initial allocation.

2. storage class

The storage class routine can assign storage classes for both data sets and objects. Objects must be SMS-managed; data sets may be SMS-managed or non-SMS-managed. Any DASD data set which is assigned a storage class is considered to be SMS-managed.

3. management class

The management class routine is run only if a storage class is assigned to the data set or object. Management classes apply only to SMS-managed data sets and objects.

4. storage group

The storage group routine is run only if a storage class is assigned to a data set. Of all the ACS routines, only the storage group ACS routine is required. The storage group assigned to a data set or object determines where the data set or object resides. All SMS-managed data sets and objects must be assigned a storage group.

All allocations directed to units that are neither tape nor DASD should be excluded from SMS management. Do this by testing for UNIT in the storage class routine and ensuring that the storage class is set to NULL in these cases. Ensuring that no storage class is assigned for such allocations avoids potential errors with allocations that require specific types of units. For example, assigning a storage class to a VTAM(®) channel-to-channel (CTC) adapter allocation results in sense errors when VTAM attempts to use the CTC.

Automatic class selection provides centralized control over data set allocation on SMS-managed volumes. If SMS is activated, all new data set allocations are subject to automatic class selection. While data class, storage class, and management class can be explicitly specified, it is the ACS routines which are the final arbiters. They may allow the use of the explicitly specified class, assign a different class, return a null value, or even fail the allocation by exiting with a non-zero code.

You must provide a storage group ACS routine because storage groups cannot be explicitly specified. The ACS routine is the only mechanism for associating storage groups with data sets and objects. This is to help remove the need for users to be aware of the physical storage environment. This in turn allows you to modify the physical storage environment without having to involve the user community.

The storage group ACS routine must return the name of a valid storage group. If it does not, the allocation will fail since the system will be unable to determine where the data set or object should be placed.

As the conversion to SMS-managed storage proceeds, you can modify the ACS routines to select more data sets to be SMS-managed. In this way, conversion to SMS-managed storage can take place gradually, minimizing user involvement and modifications to JCL or other allocation statements. You can also use DFSMSdss to convert existing data sets to SMS-managed storage without data movement.

Translating ACS Routines

After creating an ACS routine, you must translate it into executable form. The translation process checks your source code for syntactic and semantic errors, generates an object form if no errors exist, and places the object form into the SCDS you specified on the translate panel. If the ACS routine that you are translating already exists in the SCDS, the new object form replaces the existing object form. To translate an ACS routine, select option 2, Translate, from the Automatic Class Selection Application Selection panel.

Testing ACS Routines

After completing your ACS routines, you can write and execute test cases using the ISMF Automatic Class Selection application. After testing the individual routines of a new or modified configuration, you can activate it with greater confidence. Note that ACS installation exits are not invoked during ACS routine testing.

If you wish to test ACS routines, you need to allocate a partitioned data set for the ACS test cases. The partitioned data set must have a logical record length of 80 or greater and a record format of F or FB. After allocating the data set, go to the ISMF Primary Option Menu and select option 7, Automatic Class Selection Application. To specify test cases, select option 4, Test, from the Automatic Class Selection Application Selection panel.

During ACS testing, only those parameters that are passed to the ACS routines at ACS time are tested. For some data processing, what is specified in the JCL is not necessarily what is passed to the ACS routines. A good example of this is how the RETPD and EXPDT parameters work: if EXPDT is specified in the JCL, during normal processing the RETPD read-only variable is set based on the EXPDT. If your ACS routine logic bases decisions on RETPD variable, and you specify EXPDT as a

test parameter, you might get unexpected test results. ACS testing is not designed to simulate all of the possible processing that might occur before ACS processing. You must take care to know what is passed to the ACS routines at ACS time to ensure that you are testing what you expect to occur on your system.

Validating the Configuration

SMS validates the entire SCDS when it is saved. You can validate the individual ACS routines of an SMS configuration after successfully translating them. You should also validate the entire configuration yourself so that you can see any error messages that result. Separate validation of the ACS routines does not produce all of the possible messages.

You can validate an entire SCDS using the ISMF VALIDATE command. To use the command, type VALIDATE on the command line of the CDS Application Selection panel and press ENTER.

Unsuccessful Validation

For its contents to become the active storage management policy for an installation, an SCDS must be valid. Activating an SCDS validates its contents and copies them into the ACDS identified by IGDSMSxx. If the SCDS is not valid, activation fails.