Table of Contents

Introduction to VSAM

Virtual Storage Access Method (VSAM) is two things. It is a type of disk data set, and an Application Programming Interface (API) for using these data sets. Or in other words, an access method, such as sequential access, or direct access.

- VSAM data sets must be stored on disk. Tape data sets cannot be VSAM.
- VSAM is included free with z/OS, z/VSE, and z/VM, though there can be differences between these operating systems.

VSAM and Other Access Methods

There are several different data set types on the mainframe, each with advantages and disadvantages.

VSAM data sets are usually used by programs to store and manage data. They cannot be directly browsed or edited from ISPF without special software like IBM File Manager for z/OS or Compuware File-AID.

Sequential

- Data sets you can edit or browse in ISPF
- It is ideal for text
- Access method: QSAM or BSAM

```
EDIT       IBMUSER.SEQ.DSET                        Columns 00001 00072
Command ===>                                          Scroll ===> CSR
****** *************************** Top of Data ****************************
000100 SMITH       JOHN           552334  15     JONES ST
000200 JONES       SANDRA         552335  23A    BROWN ROAD
000300 GREEN       DAVID          552336  1      ALBERT LANDING
000400 OAK         ANGELA         552337  14     JONES STREET
****** ************************* Bottom of Data **************************
```

Partitioned

- A data set that has been divided into partitions called members.
- Each member contains sequential information such as data records, JCL statements or program code
- Access method: BPAM

```
EDIT            IBMUSER.PDS.DSET                   Row 0000001 of 0001177
Command ===>                                          Scroll ===> CSR
         Name    Prompt      Size  Created       Changed            ID
_____  AARINJCL            27   2014/04/08   2014/04/29 07:41:18  IBMUSER
_____  AATEST             10   2014/04/08   2014/04/15 20:54:07  IBMUSER
_____  AATEST2             1   2014/04/15   2014/04/15 22:00:26  IBMUSER
_____  ADFDOLDX           85   2014/04/08   2014/04/15 18:46:21  IBMUSER
_____  ADRDYXS1            6   2014/04/08   2014/04/15 20:03:09  IBMUSER
_____  AGHCOP             15   2014/04/08   2014/05/08 07:28:22  IBMUSER
```

VSAM

- One or more connected data sets
- It is ideal for storing data
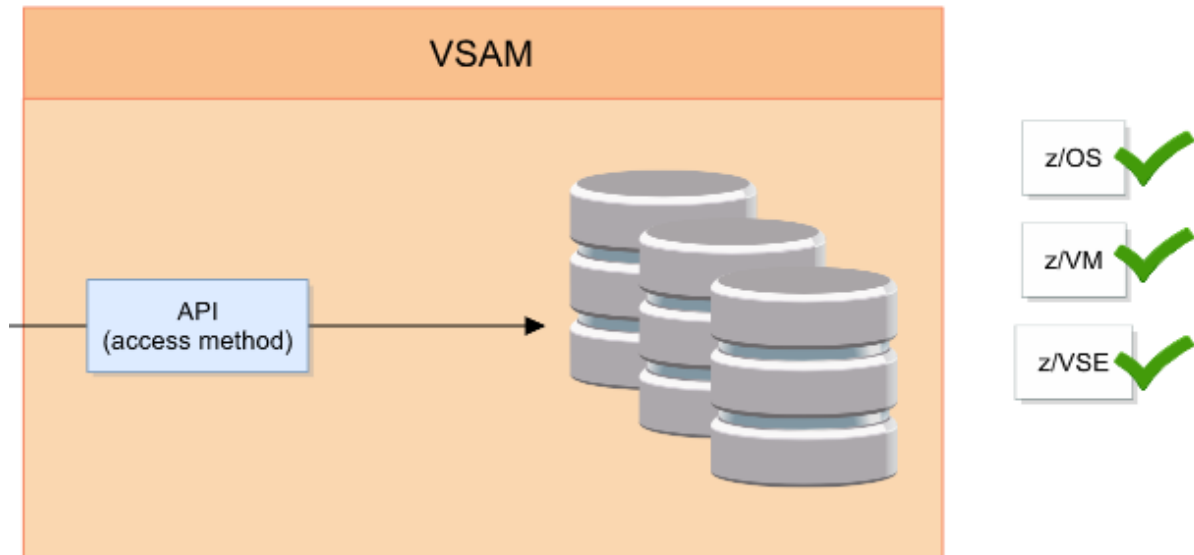
VSAM Features

VSAM provides extra features that make it excellent for storing data. These include the following:

- **Speed** - VSAM data sets are the fastest data sets available for most data-related needs. They are excellent for heavy workloads.
- **Size** - VSAM data sets can be quite big; with the introduction of extended addressability, a DB2 VSAM data set, for example, could yield a maximum size of128TB.
- **Extents** - VSAM data sets can have up to 123 extents on each disk, up to a maximum of 255 extents. They can have even more if configured by the systems admin.
- **Access** - VSAM data sets can be accessed in three ways:
  - Sequential access - one record at a time from the beginning of the data set
  - Direct access - using a key or location to directly access a record
  - Skip-sequential access - locating on record directly and then processing later records sequentially from this record

VSAM Extended

VSAM data sets have been used since the 1970s. More recently, IBM introduced a newer format of VSAM data set; VSAM extended (Extended Format VSAM).

Extended Format VSAM is a VSAM data set with extended features that must be managed by DFSMS. Application programs see no difference between VSAM and Extended Format VSAM. However Extended Format VSAM provides some advanced features for the storage administrator.

- **Sharing** - Extended format VSAM data set records can be shared by many different users at the same time, even from different systems. This is called record level sharing (RLS).
- **Size** - Prior to DFSMS 1.6, a 4 gigabyte architectural limit for data set size was imposed. With the introduction of extended addressability and extended format VSAM data sets, this maximum size was increased to 128 terabytes.
- **Compression** - Extended format VSAM data sets can be stored in a compressed format on disk.
- **Speed** - Extended format VSAM provides features to improve VSAM performance, including intelligent buffering and data set striping.

VSAM Users

VSAM is used by many different software products, and even z/OS itself. Listed below are a few examples of this:
- **IMS** - Stores some IMS databases as VSAM data sets.
- **DB2** - Uses VSAM to store Db2 objects.
- **z/OS** - Many system data sets including catalogs and SMF data sets are VSAM.
- **CA ACF2 and RACF** - User information and security rules are stored in VSAM data sets.
- **Other z/OS File Types** - z/OS uses VSAM to implement other file types such as HFS, zFS and PDSE.

VSAM Data Sets

ESDS

There are five basic types of VSAM data set. The simplest is called the Entry Sequenced Data Set (ESDS). ESDS data sets are similar to normal QSAM data sets, and can hold fixed or variable length records in any order.

ESDS data sets can be accessed sequentially or directly. They cannot be accessed skip-sequentially.

ESDS data sets are excellent for applications such as logs where sequential access is needed.

How ESDS Data Sets Work
- **Structure** - ESDS data sets hold fixed or variable length records in any order.
- **Sequential access** - An ESDS record can be located sequentially by checking each record, starting from the beginning of the data set.
- **Direct access** - An ESDS record can be located directly if the location, known as the relative byte address (RBA) is known. The location could have been determined from a previous sequential search, or by remembering the location when the record was first inserted.
- **New Records** - They must be added at the end of the data set
- **Removing Records** - Records cannot be physically deleted from an ESDS. Records that are no longer needed can be marked as deleted, but will remain in the data set until the ESDS is completely rebuilt.
- **Updating Records** - Records can be updated where they are, providing the length remains the same. If the length changes, the record must be marked as deleted, and a new record added at the end.

Control Intervals

A VSAM control interval (CI) is the basic building block of a VSAM data set. It holds one or more VSAM records.

When a record in a VSAM data set is read, it must be moved into memory from a DASD device. Similarly, when a record is written, it is moved from memory to a DASD device.
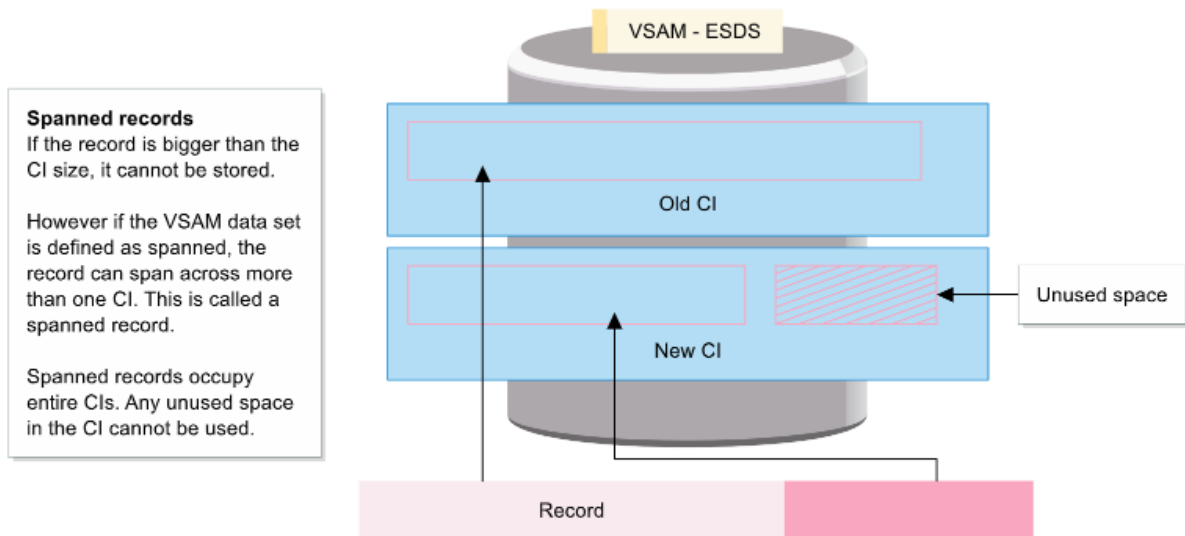- When a record is read from a VSAM data set on disk, the entire control interval is moved.

For VSAM data sets, the entire CI is moved, not just one record. This can speed up VSAM processing. The CI is similar to the block for sequential and partitioned data sets. The size of the control interval is defined by the user when creating the VSAM data set.

CIs and ESDS

For ESDS data sets, each record is inserted at the end of the data set, at the end of the most recent CI. If there is not enough room at the end of the CI, a new one is started.

If a record is larger than the CI size, it can use more than one CI, providing the VSAM data set is defined as spanned.

Spanned records
If the record is bigger than the CI size, it cannot be stored.

However if the VSAM data set is defined as spanned, the record can span across more than one CI. This is called a spanned record.

Spanned records occupy entire CIs. Any unused space in the CI cannot be used.

CI Structure

VSAM not only stores records in the CI. It also stores other information needed to manage the CI. This information includes the following
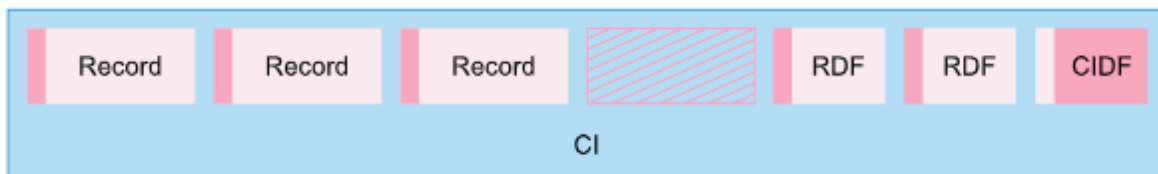
- **Records** - Data records are stored in each interval.
- **Unused Space** - For ESDS data sets, this cannot be used unless it is after the final record in the data set.
- **Record Definition Field (RDF)** - The RDF stores the length of each record. There is one per record for VSAM data sets with variable length records. VSAM data sets with fixed length records only need two; one for the record length and one recording the number of records.
- **Control Interval Definition Field (CIDF)** - This is one 4-byte field holding the size and location of unused space in the CI.

Control Areas

VSAM CIs are stored on disk in groups. These groups are called control areas (CAs). One VSAM data set can have one or more CAs.
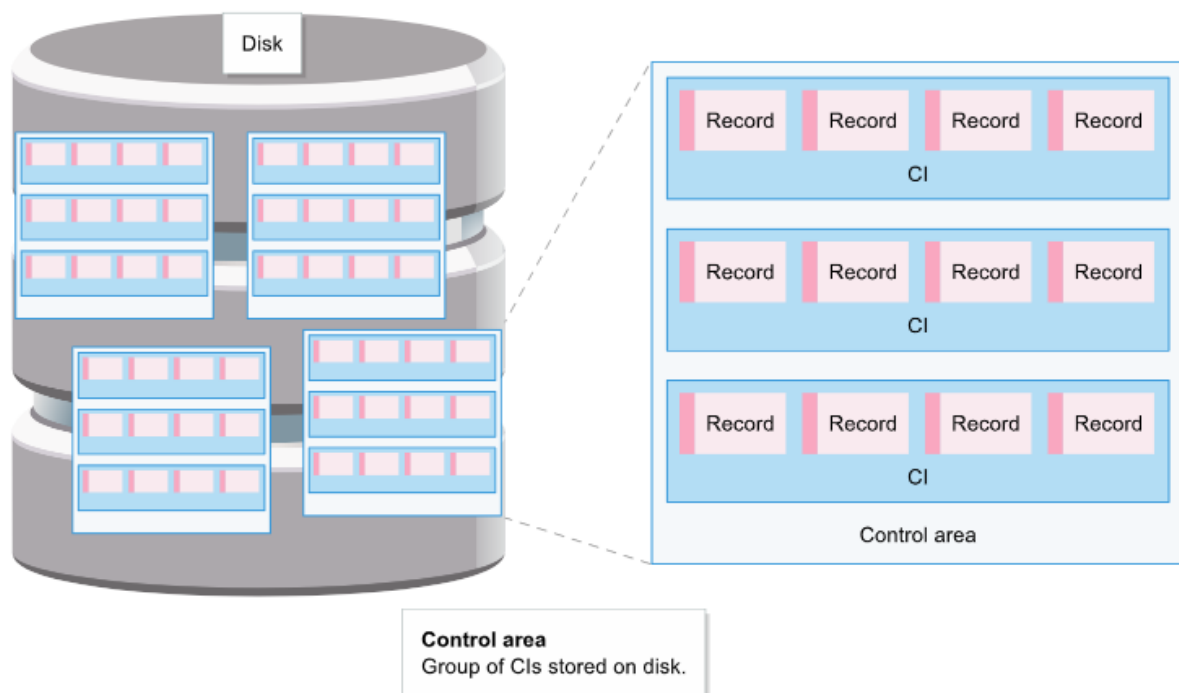
The size of the CA is determined by the system when the data set is defined.

RRDS

The relative record data set (RRDS) is another type of VSAM data set.

How RRDS Data Sets Work

- **Structure:**
    - RRDS data sets consist of fixed length areas called slots. Slots are pre-formatted when the data set is created, or whenever a new CA is created.
    - Records are inserted and deleted into these slots.
    - They also have the same CI structure as ESDA data sets, including RDFs and CIDFs. Any unused space created as a result of CI fragmentation cannot be used to store data.
    - There is one RDF for every slot; each RDF is used to indicate whether the corresponding slot is being used or is empty.
- **Limitations:**
    - All records must be the same length.
    - No spanned records; records cannot span slots or CIs.

RRDS Access

RRDS data sets have some advantages over ESDS data sets:
- Records can be added and deleted within the data set.
- Records can be directly accessed by specifying the slot number. This is called a Relative Record Number, or RRN. The first slot has an RRN of 1.
- Skip-sequential processing can be used.

RRDS data sets are excellent for applications with fixed length records where direct access by relative record numbers is needed. In reality, RRDS data sets are rarely used today.

How to access:
- **Sequential access** - Like ESDS data sets, records can be located sequentially by checking each record, starting from the beginning of the data set.
- **Direct access** - A RRDS record can be located by specifying the RRN; for example to access the 8th record the RRN would be 8.
- **Skip-sequential access** - A RRDS record can be located directly by specifying the RRN, and subsequent records accessed sequentially.
- **New Records** - These can be inserted into any free slot. The application program can specify the destination slot; this is known as direct insertion. Or the application program can request the record be inserted in the next slot; this is known as sequential insertion.
- **Removing Records** - Any record can be deleted, and the slot reused.

Linear Data Sets

Linear data sets are different from other VSAM data sets. They do not hold records, but simply strings of data.
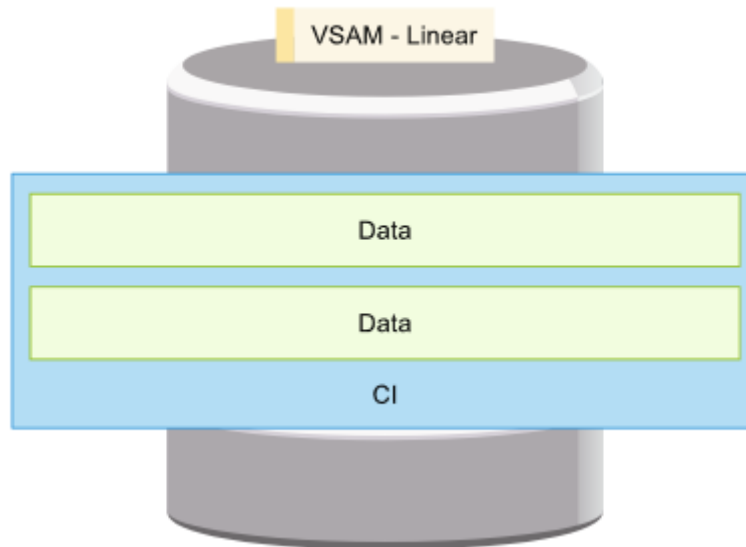
Linear data sets are usually used by programs and applications that want to specify their own internal record structure. Db2 uses linear data sets to store Db2 objects.

How Linear Data Sets Work

- **Structure** - Linear data sets hold only data. There are no records, and no RDFs or CIDFs. Programs accessing the data set must internally determine their own record structure.
- **Access** - Programs use z/OS data-in-virtual (DIV) or similar services to insert, update, read and delete data as if it was memory.

VSAM Indexes

ESDS and RRDS VSAM data sets are excellent for sequential access. However, to directly locate a record inside these data sets a program must know either the RBA for ESDS, or the RRN for RRDS.
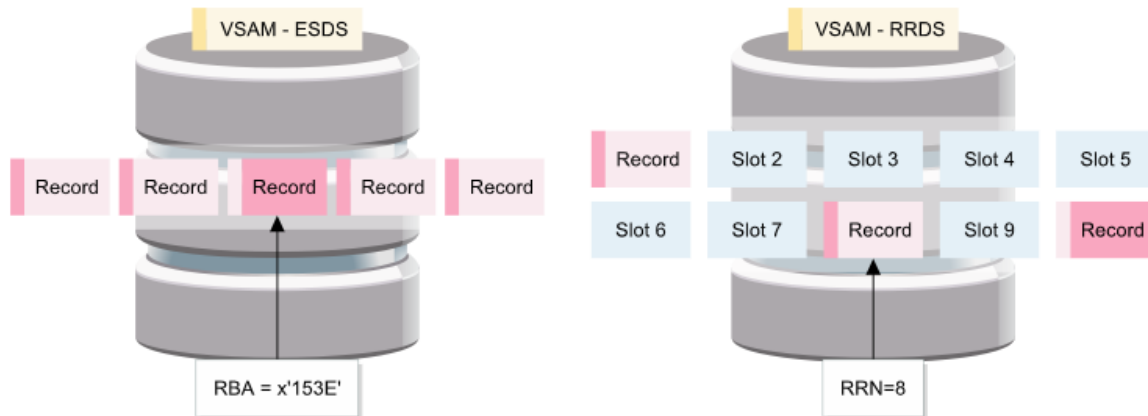
Although there are ways to obtain the RBA or RRN, it can be difficult. VSAM solves this problem with another VSAM data set type; the key sequenced data set (KSDS).

KSDS

A KSDS is actually two VSAM data sets that are used together. Each data set is called a component.
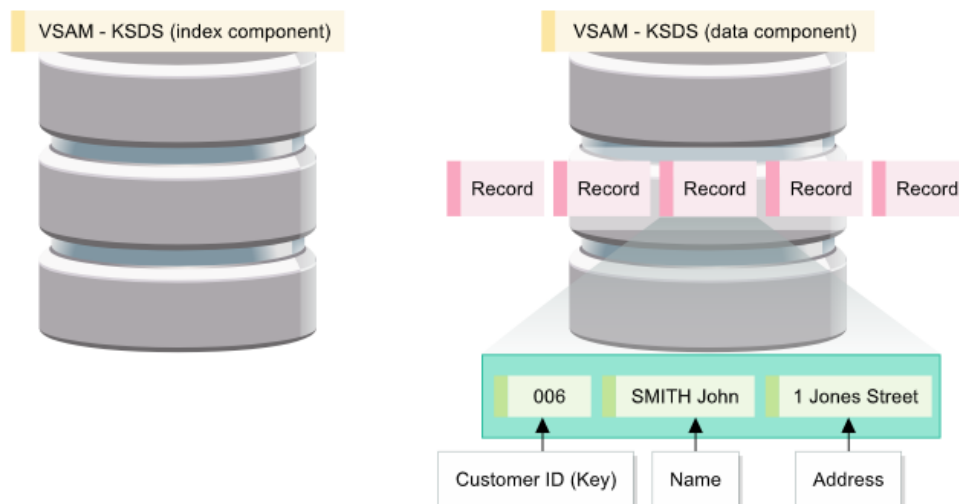
The data component holds the data in the same way as an ESDS; fixed or variable length. A part of each record is the record's key.
- This key can be any 1-255 contiguous bytes in the record that are unique.
- There cannot be two records with the same key.
- This key must also be the same length and position for all records.
- This key can be used to locate the record.

Record Insert into KSDS

Records in the data component of a KSDS are stored in key order.

They can be stored sequentially or directly.

Once a record has been inserted, its key cannot be altered. To change a record's key, the record must be deleted and re-inserted.

- **Free Space** - When defining VSAM data sets (other than RRDS), the amount of space to leave free in every CI is specified: the free space. RRDS data sets do not have free space.
- **Sequential insert** - KSDS records can be inserted sequentially. This often happens when the data set is first created and the data is loaded in. Records are inserted in key order until the free space limit is reached or the CI is full. Records are then inserted in the next CI.
- **Direct insert** - Records can also be inserted directly. This is the normal processing when using a VSAM data set that has already been loaded with data. A record is inserted in a space so that it is in key order. If there is not enough room to insert a record, other records are moved into the free space to make room for the new record.

Record Update in KSDS

Records in a KSDS data component can be updated and their length changed.

- **Same Length** - If a record is updated and the length is the same, the processing is the same as for an ESDS or KSDS.
- **Reduced Length** - If the length of the record being updated is reduced, then the remaining space is now free space. It can be reused as necessary for record inserts or updates.
- **Increased Length** - If the length of the record being updated is increased, then the records are moved into the free space to make room for the extra length.

CI Split

If there is not enough free space in a CI for a new record or increased length of an existing record, the CI is split into two. This is called a CI split.

Suppose a new record is ready to be inserted, or an existing record will be updated and its length increased. However, there is not enough free space in the CI for the new data to fit.

- The CI is split into two.
- The second half is moved to a free CI in the CA
- There is now enough free space to insert the new record, or to expand an existing one.
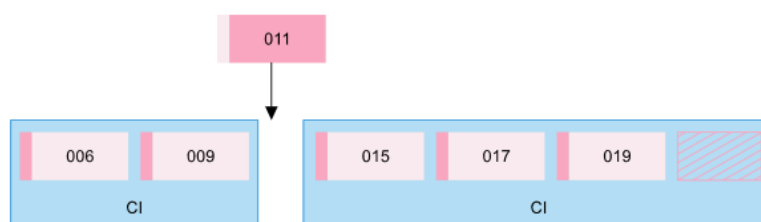
CA Split

In a CI split, half of the CI is moved to a spare CI in the CA. Like CI, the amount of space to leave free in every CA can be specified when defining the VSAM data set - the free space.
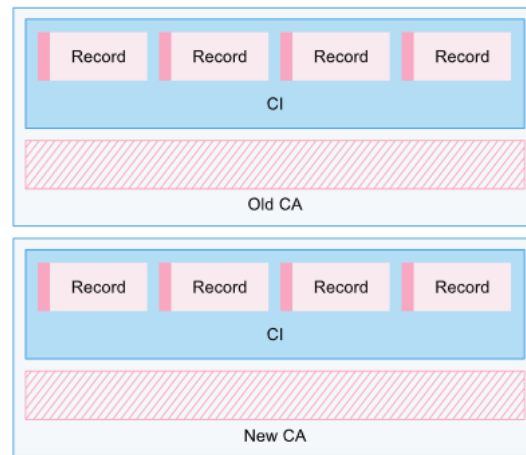
During sequential inserts, this free space is not used. If there is a CI split, one half of the CI is moved into a new CI in the CA free space.
   - If there is not enough space in the CA for another CI, then the CA is split into two.
   - This is when one half of the cold CA is moved to a new CA at the end of the VSAM data set.
   - The CI split can now be processed.

KSDS Index

The index component of a KSDS holds exactly that, an index into the data component. This is sometimes called the prime index. It is built automatically as records are inserted, updated, or deleted. It is also updated automatically for any CA or CI splits.

This index allows VSAM to locate any data record given its key value.

Each index record holds the highest possible key of a data CI, and the location (RBA) of that CI. The highest possible key is compressed to save space.
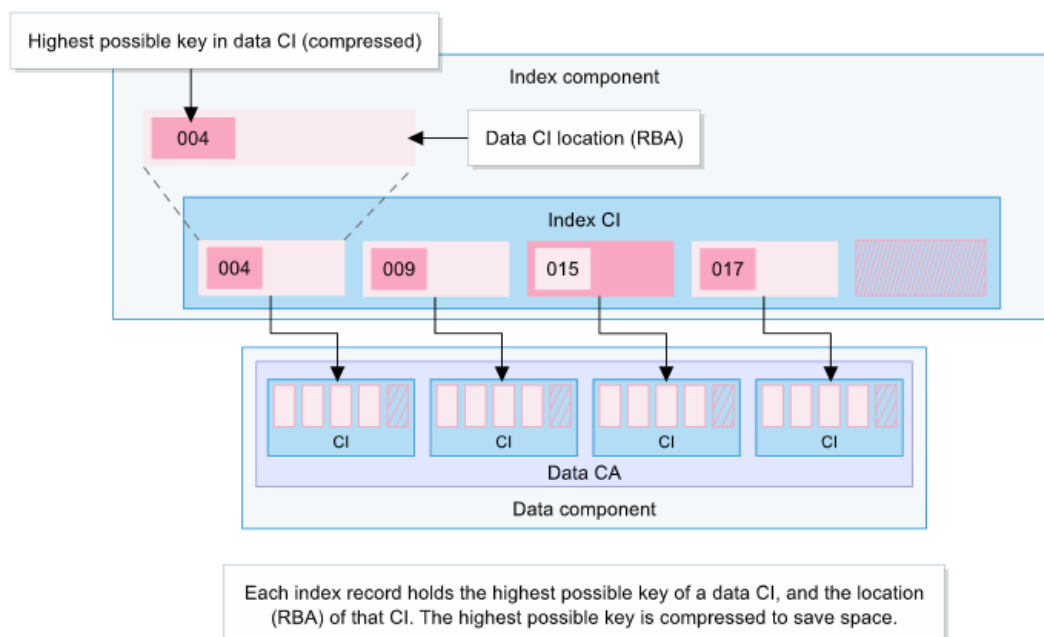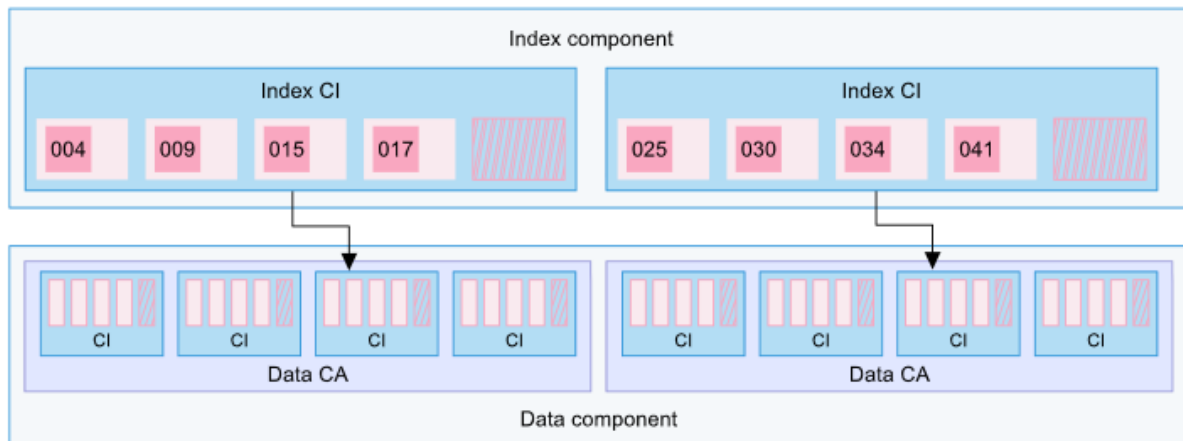
Every index CI holds all the indexes for one data CA.
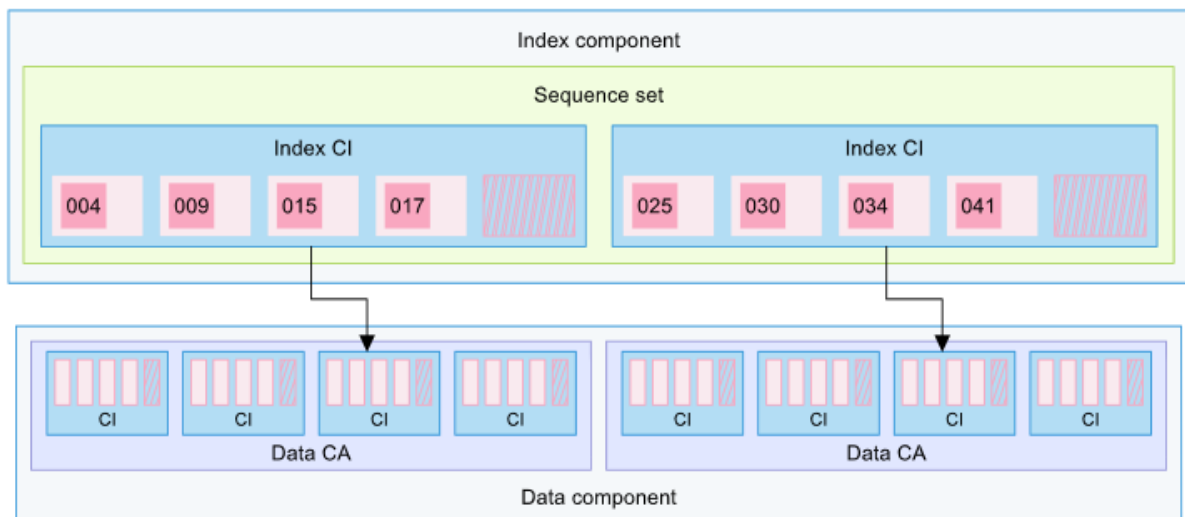
This index is called the sequence set.
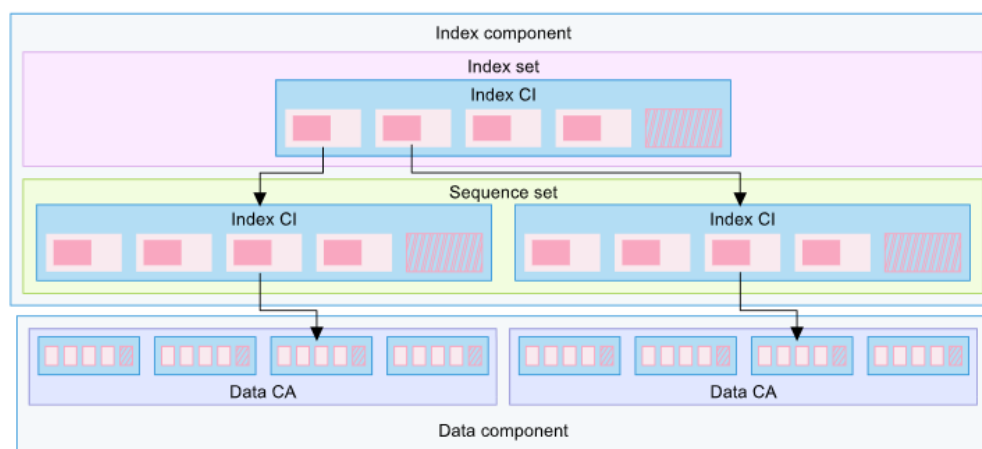
Larger VSAM data sets will have a second index level, where each record points to a sequence set CI. When accessing a VSAM data set, this second level is accessed first to find the sequence set index record. The sequence set record is then accessed to locate the data record.

Index set
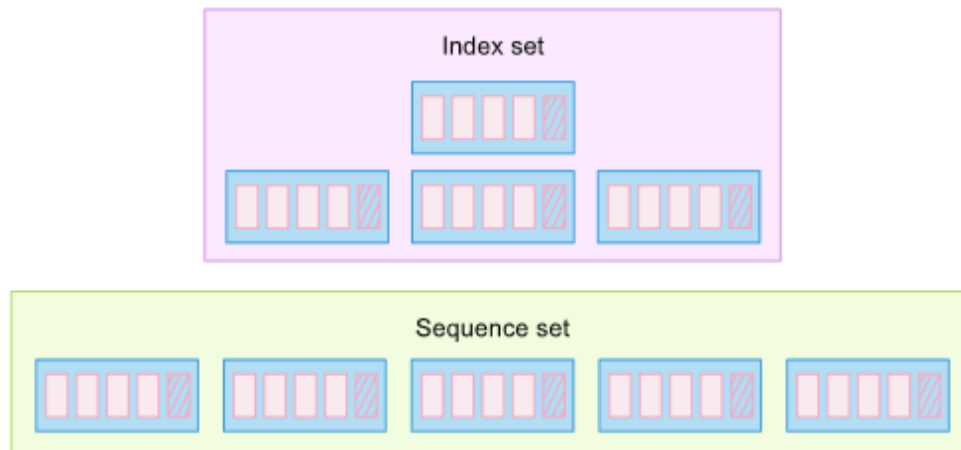
Sequence set

VSAM data sets can have many index levels. However, there will always be only one sequence set.

Index Component Structure

The KSDS index component is very similar in structure to the data component.

- **CIs and Free Space** - The index component is broken into CIs. The size of this CI can be determined when creating the VSAm data set. The free space of each index CI can also be specified when creating the VSAM data set.
- **CAs and Free Space** - Like the data component, the index CIs are grouped into CAs. Like all CAs, the size is determined by the system. The free space of each index CA can be specified when creating the VSAM data set.
- **CI Split** - Index CIs will split if there is not enough space in the CI for another index record.
- **CA Split** - If the index CA does not have enough space for a new index CI, a CA split will occur.

KSDS Data Set Names

This screen shows the ISPF DSLIST (option 3.4) listing of a VSAM KSDS. All these components together are called a VSAM cluster.

The name of each component is specified when the VSAM data set is created, and can be any valid data set name. However, most users use the cluster name for the data and index components, ending each with something like DATA or INDEX respectively.

The Components:
- **Cluster Component** - A component that relates the data and index components together. The cluster component is not a data set, only a catalog entry. All VSAM data sets have a cluster component, and any program or utility accessing a VSAM data set refers to the cluster component.
- **Data Component** - Every VSAM data set has a data component
- **Index Component** - Only KSDS and VRRDS data sets have an index component

KSDS Processing

KSDS data sets can be accessed sequentially, directly, or skip-sequentially.
- **Sequential access** - Like ESDS and RRDS data sets, records can be located or processed sequentially by accessing each record. Records are accessed from the lowest key to the highest key. Records in a KSDS are not necessarily next to each other, particularly if a CI or CA split has occurred. So sequential processing locates each record using the sequence set in the index component.
- **Direct access** - Records in a KSDS can be located directly by specifying the record key. The KSDS index is accessed to locate the record with this corresponding key.
- **Skip-sequential access** - A record in a KSDS can be located directly by specifying the record key. Records after this record are processed sequentially.

Alternate Indexes

Suppose a VSAM record can be accessed by two different keys. For example, a telephone directory may have a unique customer number and telephone number.

VSAM allows this using an alternate index. An alternate index is the way of accessing a VSAM data set using a key that is different from the defined key. So in the example, the VSAM key could be the telephone number and the alternate index is the customer number.
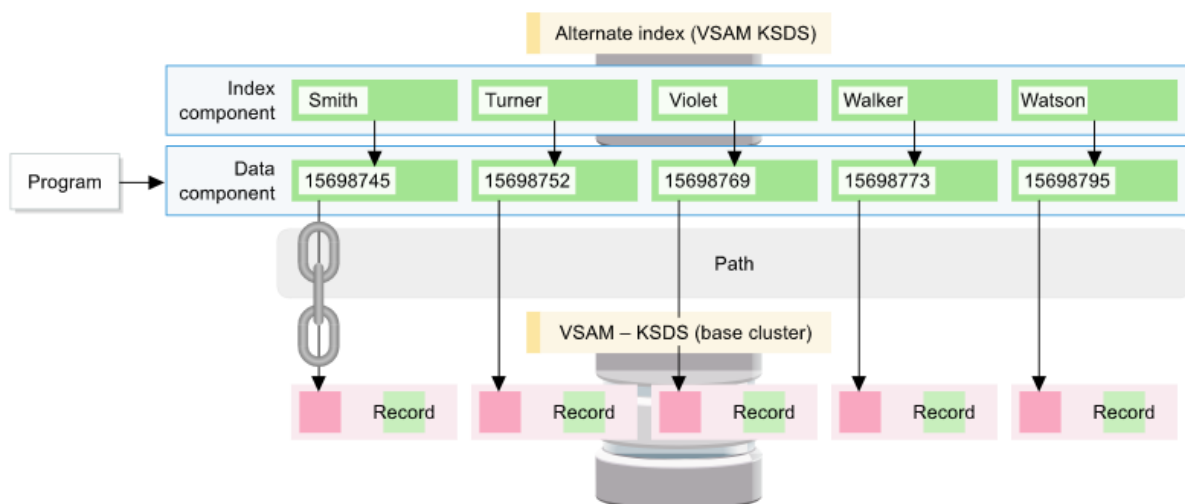
Alternate indexes can also be used to add an index to an existing ESDS data set.
- An alternate index is a group of pointers into a VSAM KSDS or ESDS data set using a different key. The VSAM data set holding the data is called the base cluster.
- For example, a KSDS may use an eight byte customer ID at the beginning of the record as the primary key. An alternate index could be defined to use the 20 byte customer name beginning at position 8 in the record.

- One base cluster can have any number of alternate indexes.
- There is one alternate index object for every alternate key in a base cluster. Like the primary key of the base cluster, alternate keys can be any 1-255 contiguous bytes in a VSAM record. However, unlike the primary key, alternate keys do not have to be unique. There can be more than one base cluster record with the same alternate key.
- The alternate index object is itself a VSAM KSDS that is created after the base cluster.
  - The data component of the alternate index holds pointers to the base cluster, either an RBA address (if the base cluster is an ESDS) or the prime key (if the data component is a KSDS).
  - The index component works the same as the index component of a VSAM KSDS.
- Once the alternate index has been built, an object called a path is created to link the alternate index to the base cluster.
- Programs access the path as if it was the base cluster, using the alternate key. The alternate index can be automatically updated by VSAM whenever the base cluster changes.

VSAM Sphere

This screen shows the ISPF DSLIST (option 3.4) listing of a VSAM KSDS together with its alternate indexes and paths. There is a base cluster component KSDS (MY.VSAM.DSET), alternate index (MY.VSAM.DSET.AIX), and path (MY.VSAM.DSET.PATH). All these components together are called a VSAM sphere.

Like the cluster components of the KSDS and alternate index, the path does not exist on disk; it is only a catalog entry that relates the base cluster and alternate index together.

All these objects can have any name specified when creating the base component, alternate index, or path.

```
DSLIST - Data Sets Matching MY.VSAM.DSET                         Row 1 of 7
Command ===>                                               Scroll ===> CSR

Command - Enter "/" to select action              Message           Volume
--------------------------------------------------------------------------------
         MY.VSAM.DSET                                                *VSAM*
         MY.VSAM.DSET.AIX                                            *AIX *
         MY.VSAM.DSET.AIX.DATA                                       VPMVSH+
         MY.VSAM.DSET.AIX.INDEX                                      VPMVSH+
         MY.VSAM.DSET.DATA                                           VPMVSH
         MY.VSAM.DSET.INDEX                                          VPMVSH
         MY.VSAM.DSET.PATH                                           *PATH*
*************************** End of Data Set list ***************************
```

VRRDS

The previous section introduced VSAM RRDS data sets, and how they must have fixed-length records.

VSAM has another data set: Variable-length Relative Record Data Set (VRRDS). Like an RRDS, the VRRDS can be accessed sequentially or directly using the RRN. However, they can hold variable length records.

Unlike RRDSs, VRRDSs do not have slots. They are in effect VSAM KSDS data sets that are accessed using an RRN instead of a key. Like KSDS data sets, they have both a data component and an index component.
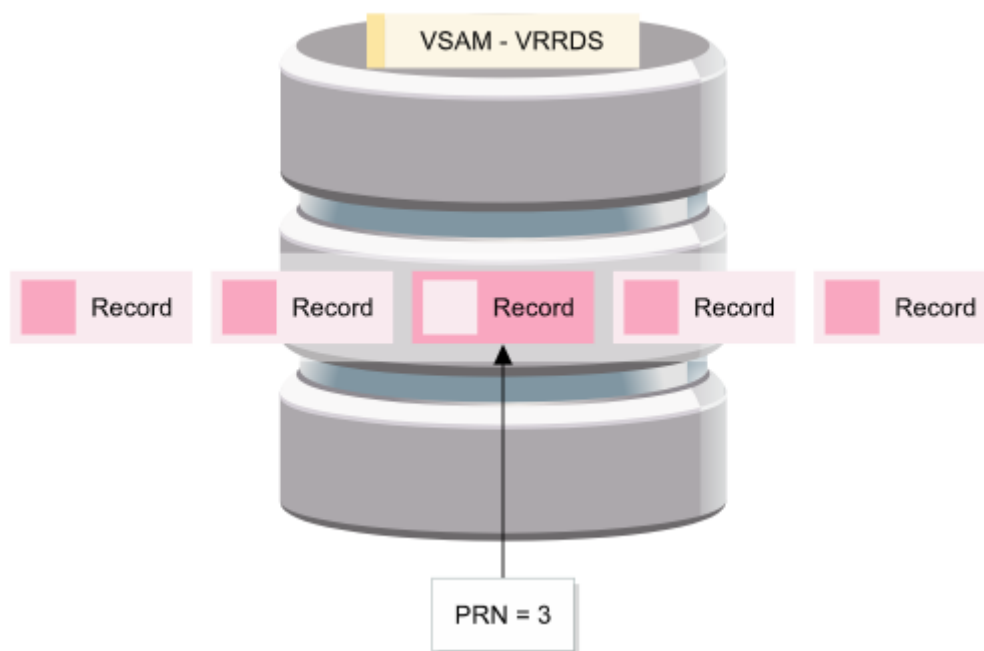
Creating VSAM Data Sets

Introduction

There are several different ways to allocate a VSAM data set; from ISPF panels and TSO/E commands to batch jobs.

Some of these include:
- **IDCAMS** - VSAM data sets can be created using the access method services utility IDCAMS from a batch job.
- **TSO/E Define** - VSAM data sets can be created using this command.
- **TSO/E Allocate** - VSAM data sets can be created using this command.
- **DD Statement** - A DD statement can be used to allocate a VSAM data set.
- ISPF Panels - The ISPF VSAM Utilities panel (ISPF option 3.2.V) can be used to create a VSAM data set.
- **Dynamic Allocate** - A VSAM can be allocated within a program using z/OS dynamic allocation services.
- **Utilities** - Utility products such as IBM File Manager for z/OS provide panels and functions to create VSAM data sets.

Using IDCAMS

The z/OS component used to create and manage VSAM data sets is called Access Method Services (AMS).

The primary AMS utility used with VSAM data sets is a batch utility called IDCAMS. IDCAMS is used frequently when dealing with VSAM data sets.

Some data set tasks:
- Create - This can create VSAM data sets, alternate indexes, and paths.

```
//JOB001 JOB (ACCT),IDCAMS,MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DEFINE CLUSTER -
     (NAME(DAVIDS.VSAM1) -
     INDEXED -
     FREESPACE(20 20) -
     VOLUMES(*) -
     STORCLAS(SC1)) -
  DATA -
     (CYLINDERS(5 1) -
     KEYS(5 0) -
     RECORDSIZE(80 80)) -
  INDEX -
     (CYLINDERS(1 1))
/*
```

- Delete - This can delete any VSAM data set or object, or any non-VSAM data sets.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   DELETE DAVIDS.VSAM1
```

- Modify - This can change VSAM data set characteristics; however, there are restrictions on what type of data set attributes can be modified.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   ALTER DAVIDS.VSAM1 NEWNAME(DAVIDS.VSAM2)
```

- Copy and Move - Can copy and move data between VSAM and non-VSAM data sets.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   REPRO INDATASET(DAVIDS.VSAM1) ⊕
         OUTDATASET(DAVIDS.QSAM1)
```

- List - Can be used to list data set information and print data stored in a data set.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   LISTCAT ENT(DAVIDS.VSAM1) ALL
```

IDCAMS Return Codes

IDCAMS returns messages describing what has happened, and a return code that indicates the success or failure of the command. Every command also returns a return code for example:
- 0 - command successful
- 4 - command with warnings
- 8 - command completed with errors
- 12 - an error prevented the command from completing
- 16 - a serious error occurred in SYSIN

Coding IDCAMS

The IDCAMS batch utility requires a SYSIN and SYSPRINT DD statement, while other DD statements may be added depending on the task being performed.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1     DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN     DD *
  /* Copy DAVIDS.VSAM1 to QSAM1 DD */
  REPRO INDATASET(DAVIDS.VSAM1) -
        OUTFILE(QSAM1)
```

Details on each statement:

- Statement 1 - The program name is IDCAMS.
- Statement 2 - IDCAMS messages and information are sent to the SYSPRINT DD; This is a required DD statement.
- Statement 3 - Other DD statements may be used by IDCAMS commands.
- Statement 4 - IDCAMS commands are read from SYSIN DD. This is a required DD statement. In this case, we are copying records from the VSAM data set: DAVIDS.VSAM1 to the sequential data set specified in the QSAM1 DD statement.
- Statement 5 - SYSIN DD statement in-stream data can contain comments by enclosing it between /* and */ strings.
- Statement 6 - SYSIN statements must be in columns 2 to 72; these margins can be changed using the MARGINS parameter on the EXEC statement.

IDCAMS Modal Commands

The IDCAMS utility is documented in the IBM z/OS DFSMS Access Method Services Command manual.

IDCAMS messages all begin with the three characters IDC, and are documented in the z/OS messages manuals or can be found by searching via the IBM Knowledge Center search facility.

Modal Commands:

- Conditional execution based on the success of the previous function:
  - In this example, the IF statement uses the LASTCC parameter to obtain the condition code from the last functional command performed (REPRO). It then compares this to a value, which if true, will invoke another command (DELETE).

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1     DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN     DD *
  REPRO INDATASET(DAVIDS.VSAM1) -
        OUTFILE(QSAM1)
  IF LASTCC = 0 THEN -
      DELETE DAVIDS.VSAM1
```

- Conditional execution based on the maximum condition code:
  - In this example, the IF statement uses the MAXCC parameter to obtain the highest condition code from any previous functional command performed within the step. It then compares this to a value, which if true, will invoke another command (DELETE).

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1    DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN    DD *
   REPRO INDATASET(DAVIDS.VSAM1) -
         OUTFILE(QSAM1)
   IF MAXCC <= 4 THEN -
      DELETE DAVIDS.VSAM1
```

- Action taken if a comparison is true or false:
  - In this example, the IF THEN and ELSE command sequence identifies the action to be taken based on the condition code from the last functional command performed (REPRO). If this is true, then the DAVIDS.VSAM1 data set will be deleted. If this is false, then the DAVIDS.VSAM2 data set will be deleted.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1    DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN    DD *
   REPRO INDATASET(DAVIDS.VSAM1) -
         OUTFILE(QSAM1)
   IF LASTCC = 0 THEN -
      DELETE DAVIDS.VSAM1
   ELSE -
      DELETE DAVIDS.VSAM2
```

- Invoking several actions based on a condition:
  - In this example, the IF statement uses DO and END to specify a group of commands (DELETE and ALTER) to be invoked if the condition code from the REPRO functional command is 0.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1    DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN    DD *
   REPRO INDATASET(DAVIDS.VSAM1) -
         OUTFILE(QSAM1)
   IF LASTCC = 0 THEN DO
      DELETE DAVIDS.VSAM1
      ALTER DAVIDS.VSAM2 NEWNAME(DAVIDS.VSAM2.COPY)
   END
```

- End processing of the current job step:
  - In this example, the IF statement uses the CANCEL command to end the current job step immediately. This means that the ALTER command will not be invoked if the condition code from the REPRO command is greater than 4.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1    DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN    DD *
   REPRO INDATASET(DAVIDS.VSAM1) -
        OUTFILE(QSAM1)
   IF LASTCC > 4 THEN -
      DELETE DAVIDS.VSAM1
   ELSE -
    CANCEL
   ALTER DAVIDS.VSAM2 NEW NEWNAME(DAVIDS.VSAM2.OLD)
```

- Reset a condition code:
  - In this example, the SET command is used to change the value of the LASTCC and MAXCC parameters to 0. This means that regardless of the result for the DELETE command, the condition code for the step will always be displayed as 0.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//QSAM1    DD DISP=OLD,DSN=DAVIDS.QSAM1
//SYSIN    DD *
 DELETE DAVIDS.VSAM1
 SET LASTCC = 0
    SET MAXCC = 0
```

IDCAMS Documentation

The IDCAMS utility is documented in the IBM z/OS DFSMS Access Method Services for Catalogs manual.

IDCAMS messages all begin with the three characters IDC, and are documented in the z/OS messages manuals.

TSO/E and IDCAMS

IDCAMS commands can also be used from TSO/E. The syntax of each IDCAMS command is the same as if it was submitted from a batch. Messages are returned to the TSO/E user.

The TSO/E HELP command can also be used to find out information about IDCAMS commands, for example, HELP DEFINE.

```
READY
DEFINE CLUSTER(NAME('DAVIDS.VSAM4') INDEXED FREESPACE(20 20) VOLUMES(*)
STORCLAS(SC1)) DATA(CYLINDERS(5 1) KEYS(5 0) RECORDS(80 80))
INDEX(CYLINDERS(1 1))
 DATA ALLOCATION STATUS FOR VOLUME VPSMSB IS 0
 INDEX ALLOCATION STATUS FOR VOLUME VPSMSB IS 0
 NAME GENERATED-(D) DAVIDS.VSAM1.DATA
 NAME GENERATED-(I) DAVIDS.VSAM1.INDEX
 STORAGECLASS USED IS SC1
 READY
```

BATCH DD

The standard JCL DD statement can be used to define and delete only DFSMS managed VSAM data sets. However, the DD statement only has a subset of the VSAM options provided by IDCAMS. For example, VRRDS data sets cannot be defined from a DD statement.

VSAM related DD statement parameters are documented in the standard z/OS JCL manuals.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IEFBR14
//DD1       DD DISP=(,CATLG),DSN=DAVIDS.VSAM3,STORCLAS=SC1,
//          LRECL=80,KEYLEN=5,KEYOFF=0,RECORG=KS,
//          SPACE=(CYL,(5,5))
```

TSO/E Allocate

The TSO/E ALLOCATE command is the standard command for allocating data sets from TSO/E. It can also be used to allocate and delete VSAM data sets.

The ALLOCATE command has the same limitations as the JCL DD statement for VSAM data sets.

VSAM related options can be found using the TSO/E HELP command HELP ALLOCATE.

```
READY
ALLOCATE DA('DAVIDS.VSAM1') NEW SPACE(5,5) CYLINDERS STORCLAS(SC1)
RECORG(KS) KEYOFF(0) KEYLEN(5) LRECL(80)
 READY
```

Dynamic Allocation

Dynamic allocation allows programs to allocate data sets. It can also be used to allocate and delete VSAM data sets.

Dynamic allocation has the same limitations as the JCL DD statement and TSO/E ALLOCATE command for VSAM data sets.

Dynamic allocation is documented in the z/OS Assembler Services Guide.

```
FILE-CONTROL.
    SELECT VSAM-DSET   ASSIGN S-PRINT  STATUS IS PRINT-STAT.

 01 ENV-VARS.
     05 ENV-NAME       PIC X(8).
     05 ENV-VALUE      PIC X(100).
     05 ENV-OVERWRITE  PIC S9(8) COMP.

  01 PRINT-STAT        PIC 99.

 PROCEDURE DIVISION.
 SETUP.
     MOVE z'PRINT' TO ENV-NAME.
     MOVE z'DSN(DAVIDS.VSAM3),NEW,CYL,SPACE(5,5),STORCLAS(SC),CATALOG
-          'LRECL(80),RECORG(KS),KEYLEN(5),KEYOFF(0)'
          TO ENV-VALUE.
     MOVE 1 TO ENV-OVERWRITE.
     CALL "setenv" USING ENV-NAME, ENV-VALUE, ENV-OVERWRITE.
     OPEN OUTPUT VSAM-DSET.
```

Focusing on IDCAMS

Define Cluster

The IDCAMS function to create a VSAM data set is the DEFINE CLUSTER command. This function can be used to specify all VSAM options.

Details:
- The DEFINE CLUSTER command is used to define the attributes of the VSAM data set and its components.
- The NAME parameter is used to specify the name of the cluster. This can be any valid data set name.
- The CYLINDERS parameter specifies the space allocation for the entire cluster, both data and index components, in cylinders. In this example, the primary space allocation is 5 cylinders, and the secondary space allocation is 1 cylinder.
  - If a secondary allocation is not required, then this parameter can be coded like this - CYLINDERS(5).
  - TRACKS, KILOBYTES, MEGABYTES and RECORDS can also be used instead of CYLINDERS to specify primary and secondary in different units.
  - A space allocation must be specified, unless it is automatically defined by DFSMS for a DFSMS managed data set.
- The VOLUMES parameter lists candidate volumes for the data set. Up to 59 volume serial numbers can be specified. In this example the VSAM data set can be multi-volume on VOL001 or VOL002. VOL001 will be attempted first, and VOL002 used if there is not enough space on VOL001.
- To allow DFSMS to choose a volume, an asterisk (*) can be used as the volume serial number - VOLUMES (*).
  - Using two or more asterisks like VOLUMES (* *) would allow the data set to be a multi-volume.
  - Volumes must be specified , unless it is automatically defined by DFSMS for a DFSMS managed data set.
- All DEFINE CLUSTER parameters must be enclosed by a set of brackets.
- When this job is submitted, the VSAM data set is created. By default VSAM uses the cluster name for the data component, adding the string '.DATA' to the end. VSAM also uses the cluster name for the index component, adding '.INDEX' to the end.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   DEFINE CLUSTER -
      (NAME(DAVIDS.VSAM1) -
      CYLINDERS(5 1) -
      VOLUMES(VOL001 VOL002) -
   )
```

Specifying Parameters

In the previous screen a VSAM data set was created, specifying the VSAM cluster name, space, and location. This will create a data set with default VSAM options.

In most cases, specific VSAM options that are different to the defaults are needed. This can be done in different ways.

Defaults:

If VSAM parameters are not specified in the DEFINE CLUSTER command, default values will be used.

Default VSAM Options
Type: KSDS
Record Length: fixed length of 4089 bytes
Spanned: no
Key Length: 64 bytes
Key Offset: 0 (starts at beginning of record)
Control Interval Size: system determined for direct access
Data component name: cluster name with '.DATA' at the end
Index component name: cluster name with '.INDEX' at the end

DFSMS Data Class:

These can optionally specify default VSAM options to be used by all VSAM data sets created with that Data Class. Storage administrators can automatically assign a Data Class to a data set, or it can be specified using the DATACLASS or DATACLAS parameter.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//SYSIN     DD   *
  DEFINE CLUSTER -
     (NAME(DAVIDS.VSAM1) -
     CYLINDERS(5 1) -
     VOLUMES(VOL001 VOL002) -
     DATACLAS(DC1) -
     )
```

DFSMS Data Class and Extended Format VSAM:

DFSMS Data Class options are the only way to create Extended Format VSAM data sets, and use EFVSAM functionality such as compression, data set striping, and system managed buffering. There are no DEFINE CLUSTER parameters to enable this functionality.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DEFINE CLUSTER -
     (NAME(DAVIDS.VSAM1) -
     CYLINDERS(5 1) -
     VOLUMES(VOL001 VOL002) -
     DATACLAS(DC1) -
     )
```

DFSMS Management Class and Storage Class:

The DFSMS management class and storage class can also be specified using the STORAGECLASS or STORCLAS and MANAGEMENTCLASS or MGMTCLAS parameters.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DEFINE CLUSTER -
     (NAME(DAVIDS.VSAM1) -
     CYLINDERS(5 1) -
     VOLUMES(VOL001 VOL002) -
     DATACLAS(DC1) -
     MGMTCLAS(MC1) -
     STORCLAS(SC1) -
     )
```

Model:

An existing VSAM data set can also be used as a template using the MODEL parameter. All VSAM options (except DATACLAS, MANAGEMENTCLASS and STORAGECLASS) not specified in the DEFINE CLUSTER command will be taken from the template, or model, data set. In the example below, all options except RECORDSIZE, CYLINDERS and VOLUMES will be taken from the data set DAVIDS.MODEL.VSAM.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DEFINE CLUSTER -
     (NAME(DAVIDS.VSAM1) -
     CYLINDERS(5 1) -
     VOLUMES(VOL001 VOL002) -
     MODEL (DAVIDS.MODEL.VSAM) -
     RECORDSIZE(80 80) -
     )
```

Data Set Structural Parameters

A number of other DEFINE CLUSTER parameters specify the structure of the VSAM data set. These include the following:

Record Size:

The RECORDSIZE parameter is used to specify the average and maximum size of records to be stored in the VSAM data set. In this example, variable length records with an average record size of 80 bytes and a maximum record size of 200 bytes will be accommodated in the data set's data component.

Keys:

The KEYS parameter identifies the prime key length and its location within the KSDS. In this example, the prime key is 5 bytes in length and begins at byte 0, which is the beginning of the data record. The KEYS parameter is not used for other types of VSAM data sets.

Data Organization:

This parameter is used to define the type of data organization for the VSAM data set.
- INDEXED - KSDS (the default)
- NUMBERED - RRDS (if fixed length records) or VRRDS (if variable length)
- NONINDEXED - ESDS
- ZFS - A linear data set used for a UNIX System Services zFS file system

Control Interval Size:

The CONTROLINTERVALSIZE or CISZ parameter specifies the control interval size for all components (data and index) in bytes. If not specified, the system chooses the CI size that is best for direct access.

Free Space:

This parameter is used to define the percentage of space to leave free for every Control Interval and Control Area when a VSAM dataset is initially loaded or a mass (or sequential) insert is performed. In this example, 20% of every CI and 30% of every CA will be left free.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   DEFINE CLUSTER -
      (NAME(DAVIDS.VSAM1) -
      CYLINDERS(5 1) -
      VOLUMES(VOL001 VOL002) -
      RECORDSIZE(80 80) -
      KEYS(5 0) -
      INDEXED -
      CISZ(23400) -
      FREESPACE(20 30) -
   )
```

Other Parameters

There are several other optional parameters that can be specified for a VSAM data set. The IBM DFSMS Access Method Services Commands manual documents them all.

USS	Commands
BUFFERSPACE	Minimum buffer space to use.
ERASE	If the catalog entry for the VSAM data set is deleted, the ERASE parameter will overwrite each VSAM data set component with binary zeros.
OWNER	Userid of the dataset owner.
REUSE	Clears the VSAM dataset every time it is opened. This is good for temporary files.
SPANNED	Allows records larger than one CI to span multiple CIs.

Sharing

VSAM data sets can be shared by different jobs and tasks, both on the same z/OS system, and other systems. The SHAREOPTIONS parameter specifies how the VSAM data set will be shared. This parameter contains two values:
- The first specifies sharing capabilities related to the same z/OS system, or another z/OS system sharing data sets using global resource serialisation.
  - 1 - Many tasks can read the data set at the same time. Only one can update it, and no users can read the data set while it is being updated. VSAM maintains data set integrity.
  - 2 - This is the same as specifying 1 except that many users can read the data set while it is being updated. VSAM maintains data set integrity, but the tasks accessing it must manage read integrity.
  - 3 - Many tasks read from and write to the data set at the same time. The tasks accessing the data set must maintain all integrity.
  - 4 - This is the same as specifying 3 except that buffers are refreshed after every update.
- The second describes how the data set will be shared across multiple systems.
  - 3 - Many tasks across the systems can be read from and written to the data set at the same time. The tasks accessing the data set are responsible for maintaining its integrity.
  - 4 - This is the same as specifying 3 except that buffers are refreshed after every update.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER -
         (NAME(DAVIDS.VSAM1) -
         CYLINDERS(5 1) -
         VOLUMES(VOL001 VOL002) -
         SHAREOPTIONS (2 3) -
  )
```

Data and Index Components

All DEFINE CLUSTER parameters mentioned so far have been for the entire cluster. Sometimes parameters may need to be specified only for a single component of the VSAM file.

Parameters for the data and index component can be specified after the Cluster components using the DATA and INDEX statements.
- Just like CLUSTER parameters, the DATA and INDEX parameters are enclosed in parentheses.
- It is common to have different parameters for the data and index component. For example, space, name, and CI size parameters are often different for each component.
- The syntax for parameters is the same in all sections of DEFINE CLUSTER. However, some parameters may not be applicable in the data or index component.
- Some parameters can be specified in either the cluster or data sections for example RECORDSIZE or KEYS.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER -
     (NAME(DAVIDS.VSAM1) -
     VOLUMES(VOL001 VOL002) -
     SHAREOPTIONS (1 3)) -
  DATA -
    (NAME(DAVIDS.VSAM1.DATA) -
     CYLINDERS(50 10) -
     CISZ(23000)) -
  INDEX -
    (NAME(DAVIDS.VSAM1.INDEX) -
     CYLINDERS(2 1) -
     CISZ(4096))
```

IDCAMS and JCL

IDCAMS has been the face of VSAM for a long time. More recently, z/OS has been enhanced to allow VSAM data sets to be created from JCL DD statements.

JCL DD statements have some advantages and disadvantages to using IDCAMS. These same advantages and disadvantages also apply to the TSO/E ALLOCATE command and z/OS dynamic allocation.

Advantages:
- Using a DD statement is the only way to create a temporary VSAM data set. Like other data sets, this is done by specifying a data set name beginning with the characters & or && or omitting the data set name.
- As the VSAM data set is created from a DD, there is no need for an extra jobstep to execute IDCAMS.
- All the VSAM definitions are in the JCL. There is no need to look in SYSIN or the IDCAMS output to see the VSAM options specified.
- Subsequent JCL steps can refer back to the DD statement creating the VSAM data set. They do not need to know the VSAM data set name.

Disadvantages:
- Not all IDCAMS parameters can be specified using a JCL DD statement; these include:
  - CONTROLINTERVALSIZE
  - ERASE
  - FREESPACE
  - REUSE
  - SHAREOPTIONS
  - SPANNED
  - WRITECHECK
  - Separate Data and Index component parameters

DATACLAS

Like IDCAMS, many VSAM options can be specified using a DFSMS data class. In particular, Extended Format VSAM options must be defined using a data class.

JCL DD statements are the same. The data class can be automatically assigned by the storage administrator, or specified on the DD statement.

```
//JOB001     JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1      EXEC PGM=MYPROG
//DD1        DD DISP=(,CATLG),DSN=DAVIDS.VSAM3,DATACLAS=DC1,
//           SPACE=(CYL,(5,5))
```

LIKE

In the previous section, you saw that the IDCAMS MODEL parameter allowed an existing data set to be used as a template or model. Unless otherwise specified, VSAM options for the model data set would be used for the new data set.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=MYPROG
//DD1       DD DISP=(,CATLG),DSN=DAVIDS.VSAM3,
//          LIKE=DAVIDS.VSAM.TEMPLATE,
//          SPACE=(CYL,(5,5))
```

In JCL, this can also be done using the LIKE parameter.

REFDD

In JCL, the REFDD parameter can be used to help model a data set for a new DFSMS managed data set by taking values from a data set defined on a previous DD statement.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=MYPROG
//DD1       DD DISP=(,CATLG),DSN=DAVIDS.VSAM3,STORCLAS=SC1,
//          LRECL=80,KEYLEN=5,KEYOFF=0,RECORG=KS,
//          SPACE=(CYL,(5,5))
//DD2       DD DISP=(,CATLG),DSN=DAVIDS.VSAM4,
//          REFDD=*.DD1
```

The syntax of the REFDD parameter is one of the following:
- *.ddname - values are obtained from a specified DD statement within the same job step.
- *.stepname.ddname - values are obtained from a specified DD statement from a previous step within the job.
- *.stepname.procstep.ddname - values are obtained from a specified DD statement that exists in a procedure of a previous step within the job.

VSAM Options

There are JCL DD parameters that can be used to define the VSAM options of a new VSAM data set. These include the following:

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IEFBR14
//DD1       DD DISP=(,CATLG),DSN=DAVIDS.VSAM1,
//          LRECL=80,KEYLEN=5,KEYOFF=0,RECORG=KS,
//          SPACE=(CYL,(5,5))
```

- LRECL - The parameter is used to specify the maximum length of records (in bytes) to be stored in the VSAM data set.
- RECORG - The parameter is used to specify the record organisation for the VSAM data set. It can be one of the following:
  - ES - ESDS
  - LS - Linear Data Set
  - KS - KSDS
  - RR - RRDS
- KEYLEN - The parameter is used to define the length of the KSDS key (in bytes)
- KEYOFF - The parameter is used to specify the location of the key within each KSDS record. In the example here, 0 indicates the first byte.

Space

Like IDCAMS, JCL DD statements can be used to specify VSAM data set space characteristics in the following units:

- Cylinders
- Tracks
- Kilobytes
- Megabytes
- Records

The primary and secondary space of the entire VSAM cluster in cylinders. In this example, the primary allocation is 5 cylinders, the secondary 2.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IEFBR14
//DD1      DD DISP=(,CATLG),DSN=DAVIDS.VSAM1,
//         LRECL=80,KEYLEN=5,KEYOFF=0,RECORG=KS,
//         SPACE=(CYL,(5,2))
```

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IEFBR14
//DD1      DD DISP=(,CATLG),DSN=DAVIDS.VSAM1,
//         LRECL=80,KEYLEN=5,KEYOFF=0,RECORG=KS,
//         SPACE=(TRK,(50,20))
```

In this example, the primary allocation for this VSAM cluster is 50 tracks with a secondary allocation of 20 tracks.

AVGREC

The AVGREC parameter can be used in conjunction with the SPACE parameter to describe the space values specified. It specifies that the first number in the space parameter is the average record length of the data set. The primary and secondary numbers are multipliers of this averager record length.

Possible values for AVGREC are as follows:

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IEFBR14
//DD1      DD DISP=(,CATLG),DSN=DAVIDS.VSAM1,
//         LRECL=80,KEYLEN=5,KEYOFF=0,RECORG=KS,
//         SPACE=(60,(500,200)),AVGREC=K
```

- U - The primary and secondary space numbers are the number of records.
- K - The primary and secondary space numbers are the number of records in thousands (multiplied by 1024).
- M - The primary and secondary space numbers are the number of records in millions (multiplied by 1024*1024, or 1048576).

Alternate Indexes

Brief Overview

Alternate indexes provide a way to access an existing key sequenced data set (KSDS), or entry sequenced dataset (ESDS), which is known as the base cluster, using a different key.
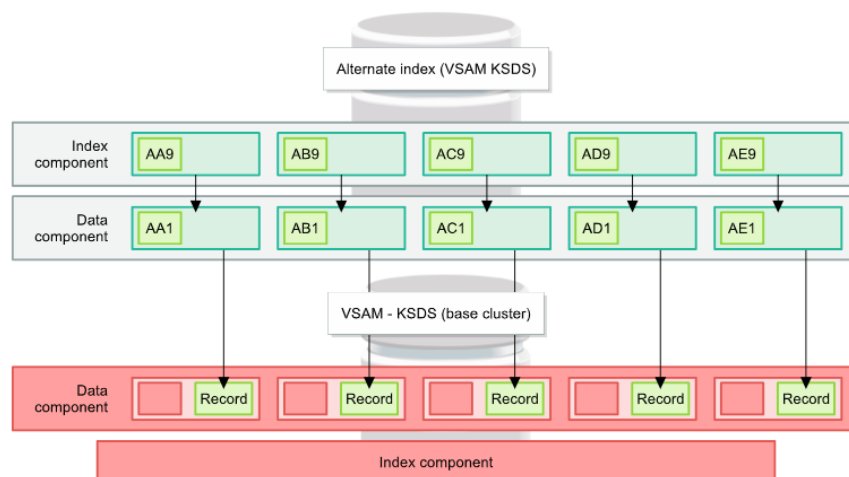
The alternative index is created after the base cluster is created and filled with data.

Creating

To create an alternative index, the alternative index object must first be created. An alternative index is itself a special type of VSAM KSDS that is created using either:
  - IDCAMS commands in batch or TSO/E.
  - The VSAM Utilities option from the ISPF Data Set Utility panel (ISPF option 3.2).

An alternative index cannot be created from a JCL DD statement, TSO/E ALLOCATE command or z/OS dynamic allocation.
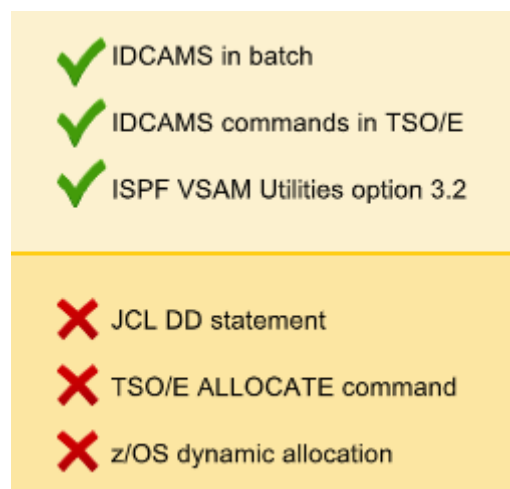
The DEFINE ALTERNATEINDEX or DEFINE AIX command is used to create an alternative index object.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DEFINE ALTERNATEINDEX -
     (NAME(DAVIDS.VSAM1.AIX) -
     VOLUMES(VOL001) -
     RELATE(DAVIDS.VSAM1)) -
  DATA -
    (NAME(DAVIDS.VSAM1.AIX.DATA) -
     CYLINDERS(5 1) -
     CONTROLINTERVALSIZE(4096)) -
  INDEX -
    (NAME(DAVIDS.VSAM1.AIX.INDEX) -
     TRACKS(15 5))
/*
```

The format of the DEFINE ALTERNATEINDEX is almost identical to the DEFINE CLUSTER command. The name of the alternative index and its components can be specified, together with space and location information.

Like the DEFINE CLUSTER command, the DEFINE ALTERNATEINDEX command can use an existing alternative index as a model or template using the MODEL parameter, or obtain VSAM options from the DFSMS data class parameters. The RELATE parameter must be specified. This names the base cluster with the data to be indexed.

Options

Alternative index options are also specified in the DEFINE ALTERNATEINDEX command.
- The KEYS parameter is used to specify the length and location of the alternate key in the base cluster. If not specified, the value (64 0) will be used. In this example, the alternate key is eight bytes long, beginning at the 21st byte of each data record in the base cluster.
- The RECORDSIZE parameter is the same as it is for DEFINE CLUSTER and specifies the average and maximum alternative index record size.
  - This alternative index record must be large enough to hold the alternate key, and one pointer (a RBA for an ESDS or primary key for a KSDS) to every base record with this key.
  - The location is the relative byte address (RBA) for an ESDS or the primary key for a KSDS.
- The UNIQUEKEY parameter specifies that the alternate key must be unique in the base cluster: it can occur in only one data record.
- The NONUNIQUEKEY parameter specifies that the alternate key can occur in more than one record in the base cluster. This is the default.
- The UPGRADE parameter specifies that the alternate index is automatically updated whenever the base cluster is updated. This is the default
- The NOUPGRADE parameter specifies that the alternate index is not automatically updated whenever the base cluster is updated. Tasks updating the base cluster must manually update the alternate index.

Create Path

A VSAM path object must be created to use an alternative index. This defines a path to the base cluster using the alternative index. The IDCAMS DEFINE PATH command is used to define a path.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 DEFINE PATH -
     (NAME(DAVIDS.PATH1) -
     PATHENTRY(DAVIDS.AIX1) -
     UPDATE)
```

- The NAME parameter specifies the path name. This can be any valid data set name.
- The PATHENTRY or PENT, parameter specifies the alternate index cluster name.
- The UPDATE parameter specifies that when the base cluster is updated using this path; all alternate indexes must all be updated. This is the default.
- The NOUPDATE parameter specifies that the alternate indexes are not updated.

Building

Once the alternative index object has been created, the alternative indexes must be built. This is done using the IDCAMS BLDINDEX command. The BLDINDEX command builds alternate index entries into an existing alternate index object.
- The INDATASET or IDS parameter specifies the base cluster name, or path pointing to the base cluster. This must be the same data set used in the RELATED keyword of the DEFINE ALTERNATEINDEX command.
- The base cluster can also be specified using a DD name. In this case, the INFILE or IFILE, parameter is used to specify the DD name.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//DD1      DD  DISP=SHR,DSN=DAVIDS.VSAM1
//SYSIN    DD  *
 BLDINDEX INFILE(DD1) -
```

- The OUTDATASET or ODS parameter specifies the cluster name of the alternate index component. The name of a path related to the alternate index can also be used. The alternate index object must be empty.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
 BLDINDEX INDATASET(DAVIDS.VSAM1) -
         OUTDATASET(DAVIDS.AIX1)
```

- Alternatively, the alternate index object or path name can be specified using a DD name. In this case the OUTFILE or OFILE parameter is used to specify the DD name.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//DD1      DD  DISP=SHR,DSN=DAVIDS.AIX1
//SYSIN    DD  *
 BLDINDEX INDATASET(DAVIDS.VSAM1) -
          OUTFILE(DD1)
```

Performance

When building an alternative index, the base cluster is read sequentially, sorted, and the alternate indexes created. This can be a large task for very big base clusters.

Other software products such as Rocket VSAM Quick Index can build alternative indexes faster than IDCAMS.

IDCAMS BLDINDEX will also attempt to improve performance by using an external sort program such as DFSORT or SyncSort's Elevate MFSort for z/OS. If these are unavailable, the internal IDCAMS sort routine will be used.

Alternate index (VSAM KSDS)

VSAM - KSDS (base cluster)

Sort
External program (DFSORT, SyncSort's Elevate MFSort for z/OS)

Build AIX
IDCAMS BLDINDEX
Rocket VSAM Quick Index

Accessing

Once an alternative index has been built, it can be accessed using the path object. The path is opened as if it was the base cluster, but the alternate key is used to locate records.
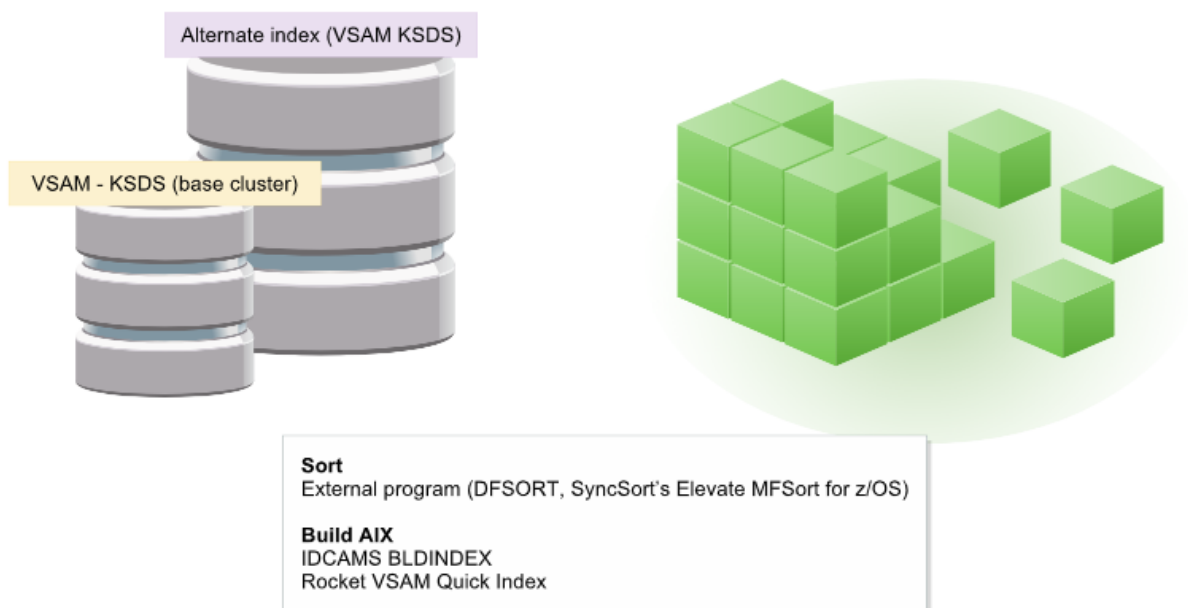
```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=MYPROG
//SYSPRINT DD SYSOUT=*
//DD1      DD DISP=SHR,DSN=DAVIDS.PATH1
```

Deleting VSAM Data Sets

Brief Overview

There are several different ways to delete a VSAM data set; from ISPF panels and TSO/E commands, to batch jobs.

Methods:
- **IDCAMS** - VSAM data sets can be deleted using the access method services (AMS) utility IDCAMS from a batch job.
- **TSO/E DELETE** - Using this command.
- TSO/E ALLOCATE - Using this with a disposition of DELETE. The data set will be deleted when the file is freed.
- **DD Statement** - Using this in JCL with a disposition of DELETE. The fill will be deleted at the end of the job step.
- **ISPF Panels** - The VSAM Utilities option from the ISPF Data Set Utility panel (ISPF option 3.2).
- **Dynamic Allocate** - Within a program using z/OS dynamic allocation services and a disposition of DELETE. The data set will be deleted when the file is freed.

IDCAMS Options

The IDCAMS DELETE command has some optional parameters that can be used.
- **ERASE** - This parameter will overwrite the deleted data set's location with zeroes for added security.
- **PURGE** The VSAM data sets can have an optional expiry data or retention period. By default, the object cannot be deleted before its expiry date. The PURGE parameter will delete the object, regardless of this expiry date.
- **NOSCRATCH** - By default, DELETE will delete the data set and its catalog entry. The NOSCRATCh parameter will delete the catalog entry, but will not delete the data set. This option is useful for deleting catalog entries with no corresponding data set, or removing tape data sets.

TSO/E DELETE

Like most IDCAMS commands, the DELETE command can also be used as a TSO/E command. The syntax of the command is the same.
- Information about the DELETE command can be seen using the TSO/E HELP DELETE command.

JCL vs IDCAMS

Like non-VSAM data sets, allocating a VSAM with a disposition of DELETE will delete the data set once it is de-allocated or freed. This can be done from a JCL DD statement, TSO/E ALLOCATE command, or z/OS dynamic allocation services.

Advantages:
- No IDCAMS Step - As the VSAM data set is deleted from a DD, there is no need for an extra jobstep to execute IDCAMS.
- No SYSIN - The processing is clear from the JCL, there is no need to look in SYSIN or the IDCAMS output to see what processing was performed.
- Programs and REXX - Deleting data sets from within a program or REXX exec is simpler.

Disadvantages:
- VSAM Clusters Only - Alternative index and paths cannot be deleted from allocation.
- Not all parameters - IDCAMS parameters cannot be specified from JCL.

Allocation

Deleting a VSAM cluster by allocation can be done using a JCL DD statement, IDCAMS ALLOCATE command, or z/OS dynamic allocation.

DD Statement:

Allocating a VSAM object with a disposition of DELETE will delete the object at the end of the job step. As with the IDCAMS DELETE statement, if the data set is a VSAM cluster, all related objects, such as the data and index components, will be deleted.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IEFBR14
//DD1       DD DISP=(OLD,DELETE),DSN=DAVIDS.VSAM3
```

TSO/E ALLOCATE:

This command can also be used to delete VSAM data sets, by specifying the DELETE parameter.

```
 READY
ALLOCATE FI(DEL1) DA('MY.VSAM.DSET') OLD DELETE
 READY
FREE FI(DEL1)
 READY
```

Dynamic Allocate:

Allocating a program using z/OS dynamic allocation services with a disposition of DELETE will delete the data set when it is freed or de-allocated.

```
PROCEDURE DIVISION.
SETUP.
    MOVE z'PRINT' TO ENV-NAME.
    MOVE z'DSN(DAVIDS.VSAM3),OLD,DELETE'
         TO ENV-VALUE.
    MOVE 1 TO ENV-OVERWRITE.
    CALL "setenv" USING ENV-NAME, ENV-VALUE, ENV-OVERWRITE.
    OPEN OUTPUT VSAM-DSET.
```

Altering VSAM Data Sets

Brief Overview

VSAM data sets have several options specifying their name, structure, and how they will be accessed.

Once a VSAM data set has been created, these options may need to be changed. This can be done without recreating the VSAM data set by using the IDCAMS ALTER command.

ALTER

The ALTER command can be used to update the options of a VSAM data set component; the cluster, data, or index components.

- The name of the component is specified after the ALTER command. This can be any VSAM object, or a non-VSAM data set.
- The name of the component can be changed using the **NEWNAME** parameter. The ALTER command will only update the component specified.
- To change the name of the cluster and all components, an ALTER command must be specified for each component separately.
- The asterisk character (*) can be used as a wildcard to alter more than one VSAM entry. Wildcards should be used with caution.
- Alter is normally not used to rename VSAM data sets. Altering a data set to a name that should reside in a different catalog can cause problems when attempting to access the data set. Normally VSAM data sets are copied to a new data set with the new name.

Cluster Options

Many VSAM options can be changed using the ALTER command. The parameters are the same as for the DEFINE CLUSTER command, with the exception of the name, which is changed using the NEWNAME parameter.

- **MANAGEMENTCLASS** - The DFSMS management class and storage class can be changed for VSAM clusters. However, the data class cannot.

```
//SYSIN    DD *
 ALTER MY.DATASETNAME MANAGEMENTCLASS(NEWMC1)
```

- **RECORDSIZE** - This parameter can be updated for cluster and data components, but only if the VSAM object being modified contains no data records and the object's current maximum record size is the VSAM default.

```
//SYSIN    DD *
 ALTER MY.DSETNAME RECORDSIZE(80 23000)
```

- **KEYS** - This parameter can be updated, but only if the VSAM data set is empty and the previous KEYS and values was the VSAM default.

```
//SYSIN    DD *
 ALTER MY.DATASETNAME KEYS(8 0)
```

- **REUSE/NOREUSE** - These parameters can be changed for clusters only.

```
//SYSIN    DD *
 ALTER MY.DATASETNAME REUSE
```

- **ERASE/NOERASE** - These parameters can be updated for data components only.

```
//SYSIN    DD *
 ALTER MY.DATASETNAME.DATA NOERASE
```

- **SHAREOPTIONS** - This parameter can be updated for data and index components only.

```
//SYSIN    DD *
 ALTER MY.DATASETNAME.DATA SHAREOPTIONS(3 3)
 ALTER MY.DATASETNAME.INDEX SHAREOPTIONS(3 3)
```

- **ADDVOLUMES** - One of the most common uses of ALTER is to add candidate volumes to a VSAM data or index component. The format of the ADDVOLUMES parameter is identical to the DEFINE CLUSTER volumes parameter.

```
//SYSIN    DD *
 ALTER MY.DATASETNAME.DATA ADDVOLUMES(*)
 ALTER MY.DATASETNAME.INDEX ADDVOLUMES(*)
```

Path and Alternate Indexes

The ALTER command can also be used to update VSAM paths and alternative indexes.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 ALTER MY.PATHNAME NEWNAME(MY.NEW.PATHNAME)
 ALTER MY.AIXNAME NEWNAME(MY.NEW.AIXNAME)
 ALTER MY.AIXNAME.DATA NEWNAME(MY.NEW.AIXNAME.DATA)
 ALTER MY.AIXNAME.INDEX NEWNAME(MY.NEW.AIXNAME.INDEX)
```

The RECORDSIZE parameter can be updated for the data component of an alternate index, but only if the alternate index is empty and the previous maximum record size was the VSAM default.

The KEYS parameter can be updated for an alternate index data component, but only if the alternate index is empty and the previous KEYS values were the VSAM default.

The UPGRADE and NOUPGRADE parameters can be changed for an alternate index.

The ERASE and NOERASE parameters can be updated for an alternate index data component.

The SHAREOPTIONS parameter can be updated for an alternate index data or index component.

As with VSAM clusters, candidate volumes can be added to the data and index components of an alternate index.

LISTCAT Command

Brief Overview

When creating a VSAM object, such as a VSAM cluster, many VSAM options can be specified. This information is stored and used for normal VSAM processing.

Often, programmers and support staff need to see this information for existing VSAM data sets.

VSAM Statistics

Statistics about VSAM data sets are also collected, including how much of the data set is allocated and used, counts of records, data set accesses, and control interval (CI) and control area (CA) splits.

This information can be valuable for managing VSAM performance, or determining VSAM data set usage.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1       DD  DISP=SHR,DSN=DAVIDS.OUT1
//SYSIN     DD *
 LISTCAT ENTRY(DAVIDS.VSAM1) OUTFILE(DD1)
```

```
READY
listcat ent('davids.vsam1')
 CLUSTER ------- DAVIDS.VSAM1
     IN-CAT --- MASTERV.CATALOG
   DATA ------- DAVIDS.VSAM1.DATA
     IN-CAT --- MASTERV.CATALOG
  INDEX ------ DAVIDS.VSAM1.INDEX
     IN-CAT --- MASTERV.CATALOG
 READY
```

Displaying VSAM Information

The IDCAMS LISTCAT command can be used to display information about a VSAM object or data set. This information is sent to SYSOUT.

The entry name is enclosed in brackets.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 LISTCAT ENTRY(DAVIDS.VSAM1)
```

The ENTRY, or ENT, parameter specifies the entry name to list.
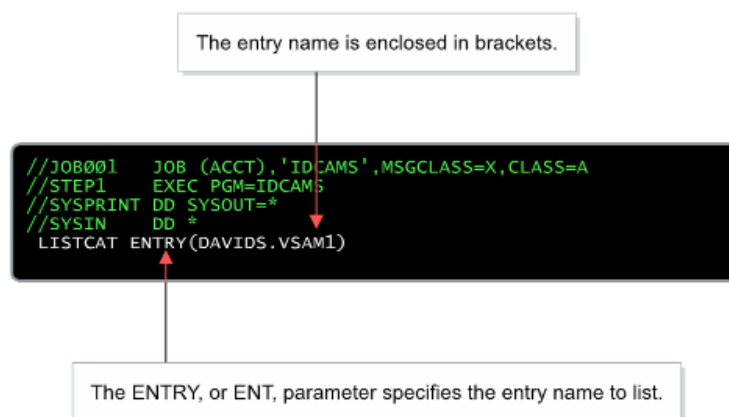
VSAM Information

Alternatively, the OUTFILE, or OFILE, parameter can be used to send the output to a different DD, as shown in the upper example.

The LISTCAT command can also be invoked as a TSO/E command, as shown in the lower example on this page. Note that the VSAM data set name is enclosed in single quotes, otherwise your TSO login ID will be used as the high-level qualifier name for the data set. The resulting output is sent to the TSO/E user.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DD1      DD  DISP=SHR,DSN=DAVIDS.OUT1
//SYSIN    DD *
 LISTCAT ENTRY(DAVIDS.VSAM1) OUTFILE(DD1)
```

```
READY
listcat ent('davids.vsam1')
 CLUSTER ------- DAVIDS.VSAM1
      IN-CAT --- MASTERV.CATALOG
    DATA ------- DAVIDS.VSAM1.DATA
      IN-CAT --- MASTERV.CATALOG
    INDEX ------ DAVIDS.VSAM1.INDEX
      IN-CAT --- MASTERV.CATALOG
 READY
```

More VSAM Information

By itself, the LISTCAT ENTRY command only outputs very basic information. To see more information, an additional parameter is required.
- The NAME parameter produces VSAM data set name information for the object. If a cluster or alternative index is specified, information for the data and index components are also displayed. NAME is the default.
- The HISTORY parameter produces the same information as NAME, plus creation and expiry dates, data set release, DFSMS data class, storage class and management class, and the latest backup date.
- The VOLUME parameter produces the same information as HISTORY, plus information on the disk volumes where the data extents reside.
- The ALLOCATION parameter produces the same information as HISTORY, plus VSAM allocation information.
- The ALL parameter produces all the information from the previous parameters, plus extra information on VSAM options and record usage. This is the most commonly used LISTCAT parameter.

```
LISTCAT ENT(DAVIDS.VSAM1) ALL
```

LISTCAT ENTRY

The LISTCAT ENTRY command will also list catalog-related information for non-VSAM data sets, GDGs, and other catalog entry types.

LISTCAT can be used for:
Alias entries
VSAM cluster
VSAM alternative index
VSAM path
Catalog
Page data set
Generation data group (GDG)
Tape library
Tape library volume entry

LEVEL Parameter

The LISTCAT ENTRY command lists information for one or more entries. For clusters and alternative indexes, information about the data and index components is also output.

- The (*) wildcard character can be used with the ENTRY or LEVEL parameter to list more than one entry and can be used to replace an entire qualifier. The perfect(%) character can be used to replace a single character with the LEVEL parameter. The asterisk wildcard cannot be used for the last qualifier. For example, MY.* is not valid.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  LISTCAT ENTRY(MY.*.VSAM1)
  LISTCAT LEVEL(MY.*.VSAM1)
  LISTCAT LEVEL(MY.H%%.VSAM1)
```

- LISTCAT LEVEL list VSAM data set information for all entries beginning with the string specified. The string must be a complete level.
    - For clusters and alternative indexes, information about the data and index components will not be output unless their names match the level of qualification.
    - In this example, information for all entries with the high level qualifier HLQ1 will be listed.

```
//SYSIN    DD *
  LISTCAT LEVEL(HLQ1)
```

- For both ENTRY and LEVEL parameters, the data output can be limited to an entry type. In this example, only VSAM clusters are output.
    - Other valid types include ALTERNATIVEINDEX, PATH, DATA and INDEX.

```
//SYSIN    DD *
  LISTCAT LEVEL(HLQ1) CLUSTER
```

Reading LISTCAT Output

Name Section

The Name section shows the entry name and type, and the catalog where it is stored. The Name section is always shown.

```
CLUSTER ------- DAVIDS.VSAM1
        IN-CAT --- MASTERV.CATALOG
```

- The Name section is at the top of the LISTCAT output for the entry. It is always output regardless of the LISTCAT parameters.
- The first field shows the type of catalog entry - valid types include cluster, data, index, path and alternative index.
- The second field shows the name of the entry.
- The third field shows the name of the catalog where this entry is stored.

History Section

The History section shows information about the data set owner, and creation and expiry dates. It is shown for most catalog entries when the HISTORY, VOLUME, ALLOCATION, or ALL parameters are specified.

```
HISTORY
   DATASET-OWNER-----DAVIDS      CREATION--------2019.123
   RELEASE---------------2      EXPIRATION------2020.365
```

- The first field shows the data set owner. By default this is the USERID of the user that created the VSAM data set.
- The second field is the data the data set was created or migrated by DFSMShsm. It is a Julian date of the format YYYY.DDD where YYYY is the year and DDD is the day (001 = January 1st).
- The RELEASE field identifies the VSAM release under which the data set was created. This is always coded as 2 (as it refers to OS/VS release 3.6 or later).
- The final field shows the Julian expiry data, or zeroes if there is no expiry date. If the expiry data is not zeros, the data set cannot be deleted before this date without using the PURGE option.

SMSDATA Section

The SMSDATA section shows DFSMS related information. It is shown for DFSMS managed clusters, alternate indexes, and non-VSAM data sets when the HISTORY, VOLUME, ALLOCATION, or ALL options are specified.

```
SMSDATA
   STORAGECLASS --------SC1      MANAGEMENTCLASS----MCDEF
   DATACLASS -------GREGTST      LBACKUP ---0000.000.0000
   CA-RECLAIM--------(YES)
   EATTR------------(NULL)
   BWO STATUS------00000000      BWO TIMESTAMP---00000 00:00:00.0
   BWO-------------(NULL)
```

- The DFSMS Storage Class, Management Class and Data Classes are displayed. These values can be (NULL) if there is no value.
- If CA-RECLAIM is YES, empty CAs will be reclaimed and reused.
- The LBACKUP field shows the date the data set was last backed up. It will be all zeros if not backed up, or 'X' characters if the date is unavailable.
- The EATTR field shows if this is an extended attribute data set: (YES) if it is, (NO) or (NULL) if not. Extended attribute data sets can reside on an extended address disk volume.
- The Backup While Open (BWO) statistics are also shown. If the BWO field is (NULL), the data set is not eligible for BWO processing.

RLSDATA Section

The RLSDATA section shows DFSMS record-level sharing (RLS) information for VSAM Clusters. It is shown when the HISTORY, VOLUME, ALLOCATION, or ALL options are specified. If the RLS IN USE field is (NO), this data set is not using RLS.

```
RLSDATA
   LOG ---------------(NULL)    RECOVERY REQUIRED --(NO)    FRLOG -----(NULL)
   VSAM QUIESCED -------(NO)    RLS IN USE ---------(NO)    LOGREPLICATE-(NO)
   LOGSTREAMID------------------------(NULL)
   RECOVERY TIMESTAMP LOCAL-----X'0000000000000000'
   RECOVERY TIMESTAMP GMT-------X'0000000000000000'
   DATABASE ----------------(NULL)
```

Encryption and Protection

The DATA SET ENCRYPTION field is (YES) if the data set has been encrypted and will also display the field DATA SET KEY LABEL with the key label specified.

```
ENCRYPTIONDATA
   DATA SET ENCRYPTION-----(NO)
PROTECTION-PSWD-----(NULL)      RACF---------------(NO)
```

The protection section shows the security for this entry. If the RACF field is (YES), this data set is protected by RACF or a similar security product. If the PROTECTION-PSWD is not (NULL), this data set is password protected. Protection information is displayed for all entries when the HISTORY, VOLUME, ALLOCATION, or ALL options are specified.

Associations Section

The Associations section will list any objects associated with the item specified in the LISTCAT command. It is displayed when the ALL option is specified.

- A cluster will display any data or index components. It will also display any associated paths and alternate indexes.

```
ASSOCIATIONS
   DATA-----DAVIDS.VSAM1.DATA
   INDEX----DAVIDS.VSAM1.INDEX
```

- A data or index component will display the parent cluster name.

```
ASSOCIATIONS
   CLUSTER--DAVIDS.VSAM1
```

- An alternative index will display all data and index components, the base cluster and any associated paths.

```
ASSOCIATIONS
   DATA-----DAVIDS.AIX2.DATA
   INDEX----DAVIDS.AIX2.INDEX
   CLUSTER--DAVIDS.VSAM2
   PATH-----DAVIDS.PATH2
```

- A path will display all components of any associated cluster or alternative index.

```
ASSOCIATIONS
   AIX------DAVIDS.AIX2
   DATA-----DAVIDS.AIX2.DATA
   INDEX----DAVIDS.AIX2.INDEX
   DATA-----DAVIDS.VSAM2.DATA
   INDEX----DAVIDS.VSAM2.INDEX
```

Attributes Section

The Attributes section shows all VSAM options. It is displayed when the ALL option is specified.

- The key length (KEYLEN) and related key position (RKP - also known as the key offset), 0 in this example, are displayed.
- If the data set is an Extended Format VSAM dataset, the EXTENDED attribute will be shown. The number of stripes will also be shown.
- If the data set is an Extended Format VSAM data set that is compressed, the COMP-FORMT attribute will be shown.
- Other VSAM options are also displayed. The format of these is the same as the IDCAMS DEFINE CLUSTER parameters.

```
ATTRIBUTES
   KEYLEN------128      AVGLRECL--2800      BUFSPACE---1024000      CISIZE---26624
   RKP----------0       MAXLRECL--3072      EXCPEXIT----(NULL)      CI/CA-------30
   STRIPE-COUNT--1
   SHROPTNS(1,3)  RECOVERY    UNIQUE      NOERASE     INDEXED     NOWRITECHK
   UNORDERED      REUSE       NONSPANNED  COMP-FORMT  EATTR       EXTENDED
```

Statistics Section

The Statistics section shows statistics about the data set activity for data and index components. It is updated every time the data set is closed, so the statistics may be outdated for data sets that are open for a long time. The data displayed depends on the entry type being displayed. It is displayed when the ALL option is specified.

```
STATISTICS
   REC-TOTAL-----------4        SPLITS-CI-------------0        EXCPS----------------21
   REC-DELETED--------14        SPLITS-CA-------------0        EXTENTS---------------1
   REC-INSERTED-------16        FREESPACE-%CI---------0        SYSTEM-TIMESTAMP:
   REC-UPDATED---------0        FREESPACE-%CA---------0              X'C8E2D37F10FD6D19'
   REC-RETRIEVED------38        FREESPC----------4765696
```

- The total number of records in the data set, and the total number of operations (EXCPS) to this data set are displayed.
- The number of record operations (such as deletions, insertions, updates and retrievals are shown).
- The number of CI and CA splits.
- The free space and number of extents.
- The timestamp when the data set was last closed in the Time of Day (TOD) or STCK format.
- Index components will show index related information, including the number of index levels.

```
EXCPS-----------------4        INDEX:
EXTENTS---------------1         LEVELS---------------1
SYSTEM-TIMESTAMP:                ENTRIES/SECT---------5
      X'C8E2D37F10FD6D19' SEQ-SET-RBA-----------0
                                HIGH-LEVEL-RBA-------0
```

Allocation Section

The Allocation section shows statistics about the total space allocation for the data or index component. It is displayed when the ALL option is specified.

```
ALLOCATION
   SPACE-TYPE------CYLINDER      HI-A-RBA---------4792320
   SPACE-PRI-------------6       HI-U-RBA----------798720
   SPACE-SEC-------------6
```

- The total primary and secondary space allocation is shown.
- The highest allocated location or relative byte address (RBA).
- The highest used location or RBA.

Volume Section

The Volume section shows information about disk volumes used by this entry. It is displayed when the VOLUME or ALL options are specified.

```
VOLUME
  VOLSER------------VPSMSD       PHYREC-SIZE--------26624     HI-A-RBA---------4792320     EXTENT-NUMBER----------1
  DEVTYPE------X'3010200F'       PHYRECS/TRK------------2     HI-U-RBA----------798720     EXTENT-TYPE-------X'00'
  VOLFLAG-----------PRIME        TRACKS/CA------------15
  EXTENTS:
  LOW-CCHH-----X'002B0000'       LOW-RBA--------------0       TRACKS---------------90
  HIGH-CCHH----X'0030000E'       HIGH-RBA---------4792319
```

- For each volume used, the volume serial number, volume type, and number of extents on that volume are shown.
- Information about the physical record size, number of records per track and the number of tracks per CA are also shown.
- Finally, the highest allocated and highest used byte locations, the RBA, on this volume are shown.
- For each extent, the location and size is shown. The disk location is low and high CCHH (cylinder and head number).
- The part of the VSAM entry residing in this extent by low and high RBA is also shown.

PRINT Command

Brief Overview

Sometimes some or all records in a VSAM data set will need to be printed or output.
IDCAMS includes the PRINT command to do exactly this.
The INDATASET parameter determines the name of the data set to print.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 PRINT INDATASET(DAVIDS.VSAM1)
```

Alternatively, the INFILE parameter can be used to specify a DD statement specifying the
input data set.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INDD      DD DISP=SHR,DSN=DAVIDS.VSAM1
//SYSIN     DD *
 PRINT -
 INFILE(INDD)
```

PRINT Output Destination

PRINT output is sent to the SYSPRINT DD. Alternatively, the OUTFILE parameter can be
used to send the output to another DD.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//OUTDD     DD DISP=OLD,DSN=OUTPUT.DSET
//SYSIN     DD *
 PRINT INDATASET(DAVIDS.VSAM1) OUTFILE(OUTDD)
```

Character Format Output

The PRINT command can produce different types of output.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 PRINT INDATASET(DAVIDS.VSAM3) CHARACTER
```

- The character format can be produced using the CHARACTER or CHAR parameter.
- For KSDS data sets, the record key is displayed in character format.
- For RRDS data sets, the location (the relative record number or RRN) is displayed as
  a decimal number.
- For ESDS data sets, the location (the RBA) is displayed as a decimal number.
- Each record is output as EBCDIC characters. Any non-printable characters appear
  as a period (.).

HEX Format Output

The HEX format prints records as hexadecimal numbers.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 PRINT INDATASET(DAVIDS.VSAM3) HEX
```

- For KSDS data sets, the record key is displayed in hexadecimal format.
- For RRDS data sets, the location (the relative record number or RRN) is displayed as a decimal number.
- For ESDS data sets, the location (the RBA) is displayed as a decimal number.
- Each record is output as hexadecimal numbers.

Dump Format Output

The DUMP format prints records in both character and dump format and is the default when no format is specified.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 PRINT INDATASET(DAVIDS.VSAM3) DUMP
```

- For KSDS data sets, the record key is displayed in hexadecimal format.
- For RRDS data sets, the location (the relative record number or RRN) is displayed as a decimal number.
- For ESDS data sets, the location (the RBA) is displayed as a decimal number.
- The offset in the record is output at the left.
- The hexadecimal format of each record is output in the middle.
- The character format of each record is output on the right.

Skipping Records

By default, the PRINT command will print records from the start of the data set. However, PRINT can also skip some records, and print a subset of the records in the data set.
- For KSDS data sets, the FROMKEY or FKEY, parameter can be used. All records after this key will be printed.

```
//JOB001    JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 FROMKEY(1110001)
```

- For ESDS and KSDS data sets, the FROMADDRESS or FADDR, parameter can be used to specify the starting RBA. All records after this RBA will be printed. This value can be a decimal number.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 FROMADDRESS(160)
```

- For RRDS data sets, the FROMNUMBER or FNUM parameter can be used to specify the starting RRN. All records after this RRN will be printed. This value is a decimal number.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 FROMNUMBER(160)
```

- The FROMNUMBER value can also be a hexadecimal number or a binary number (e.g. B '1100000')

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 FROMNUMBER(X'A0')
```

- The SKIp parameter skips over the number of records specified. In this example, all records from the 30th record are printed.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 SKIP(29)
```

- Only one of the following options above can be specified.

Limiting Records Printed

The PRINT command will print records to the end of the data set. However, PRINT can also stop printing before the end of the data set
- For KSDS data sets, the TOKEY or TKEY parameter can be used. All records appearing up to, and including this key will be printed.
- For ESDS and KSDS data sets, the TOADDRESS or TADDR, parameter can be used to specify the ending RBA. All records appearing up to, and including this RBA will be printed.This value is the same format as FROMADDRESS.
- For RRDS data sets, the TONUMBER or TNUM parameter can be used to specify the ending RRN.All records appearing up to, and including this RRN will be printed. This value has the same format as FROMNUMBER.
- The COUNT parameter prints the number of records specified. In this example, the first 30 records are printed.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 COUNT(30)
```

- Only one of the following options above can be specified.
- However, TOKEY, TOADDRESS, TONUMBER and COUNT can be used with FROMKEY, FROMADDRESS, FROMNUMBER and SKIP to print a part of a VSAM data set.

```
//JOB001   JOB (ACCT),'IDCAMS',MSGCLASS=X,CLASS=A
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 PRINT INDATASET(DAVIDS.VSAM1) -
 FROMKEY(1110000001) COUNT(30)
```

Sequential Data Sets

The IDCAMS PRINT command can also be used to print sequential data sets and PDS members.

The command format and output is the same as for VSAM data sets. However, only the COUNT and SKIP parameters can be used to limit the records printed.

```
LISTING OF DATA SET -DAVIDS.SEQ1
RECORD SEQUENCE NUMBER - 1
1110000000JONES          JOHN        1 SMITH STREET        SUBURB TOWN    00070000
RECORD SEQUENCE NUMBER - 2
1110000001SMITH          ROBERT      5 GREEN STREET        SUBURB TOWN    00080000
RECORD SEQUENCE NUMBER - 3
1110000002BROWN          JANE        18 BLUE STREET        SUBURB TOWN    00090000
IDC0005I NUMBER OF RECORDS PROCESSED WAS 3
```