VSAM - QUICK GUIDE

VSAM - OVERVIEW

Virtual Storage Access Method *VSAM* is high performance access method and data set organization, which organizes and maintains data via a catalog structure. It utilizes virtual storage concept and can protect datasets at various levels by giving passwords. VSAM can be used in COBOL programs like physical sequential files. VSAM are the logical datasets for storing records. Files can be read sequentially and randomly in VSAM. It is an improved way of storing data which overcomes some of the limitations of conventional file systems like Sequential Files.

Characteristics of VSAM

Following are the characteristics of VSAM –

- VSAM protects data against unauthorized access by using passwords.
- VSAM provides fast access to data sets.
- VSAM has options for optimizing performance.
- VSAM allows data set sharing in both batch and online environment.
- VSAM are more structured and organized in storing data.
- Free space is reused automatically in VSAM files.

Limitations of VSAM

The only limitation of VSAM is that it cannot be stored on TAPE volume. It is always stored on DASD space. It requires a number of cylinders to store the data which is not cost-effective.

VSAM - COMPONENTS

VSAM consists of following components –

- VSAM Cluster
- Control Area
- Control Interval

VSAM Cluster

VSAM are the logical datasets for storing records and are known as clusters. A cluster is an association of the index, sequence set and data portions of the dataset. The space occupied by a VSAM cluster is divided in contiguous areas called Control Intervals. We will discuss about control intervals later in this module.

There are two main components in a VSAM cluster –

- **Index Component** contains the index part. Index records are present in Index component. Using index component VSAM is able to retrieve records from the data component.
- **Data Component** contains the data part. Actual data records are present in Data component.

Control Interval

Control Intervals CI in VSAM are equivalent to blocks for non-VSAM data sets. In non-VSAM methods, the unit of data is defined by the block. VSAM works with logical data area which is known as Control Intervals.

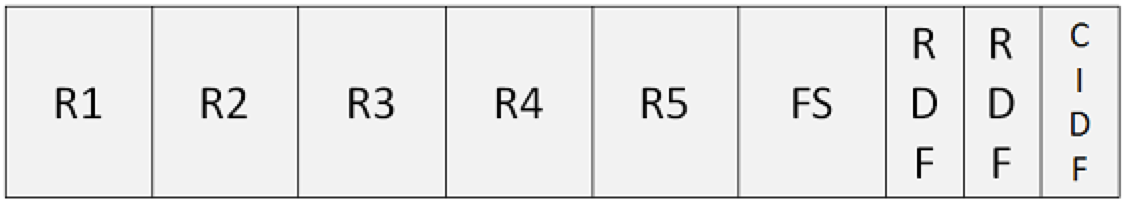
Control Intervals are the smallest unit of transfer between a disk and the operating system. Whenever a record is retrieved directly from the storage, the entire CI containing the record is read into VSAM Input-Output buffer. The desired record is then transferred to work area from VSAM buffer.

Control Interval consists of –

- Logical Records
- Control information fields
- Free Space

When a VSAM dataset is loaded, control intervals are created. The default Control Interval size is 4K bytes and it can extend up to 32K bytes.

Analysis of Control Interval



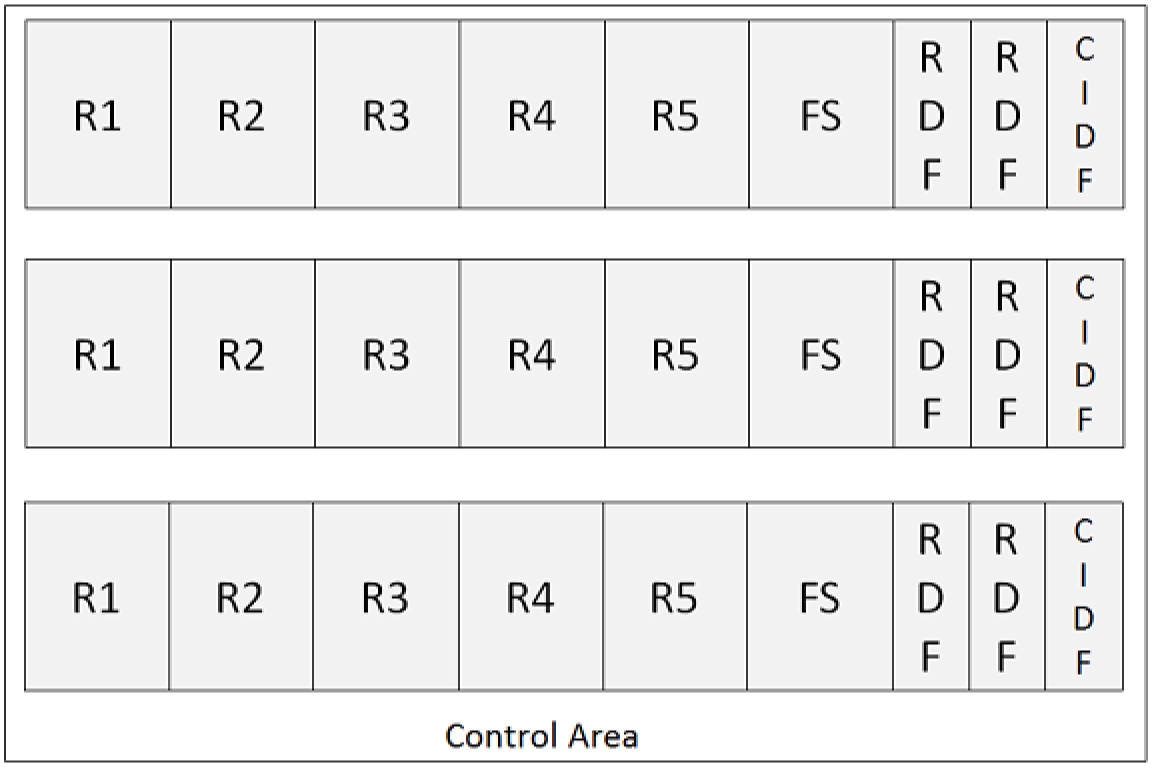
Following is the description of terms used in above program –

- **R1..R5** – Records which are stored in Control Interval.
- **FS** – FS is free space, which can be used for further expansion of dataset.
- **RDF** – RDF is known as Record Definition Fields. RDF are 3 bytes long. It describes the length of records and tells how many adjacent records are of the same length.
- **CIDF** – CIDF is known as Control Interval Definition Fields. CIDF are 4 bytes long and contain information about the Control Interval.

Control Area

A Control Area CA is formed by putting together two or more Control Intervals. A VSAM dataset is composed of one or more Control Areas. The size of VSAM is always a multiple of its Control Area. VSAM files are extended in units of Control Areas.

Following is the example of Control Area –



VSAM - CLUSTER

VSAM cluster is defined in **JCL**. JCL uses **IDCAMS** utility to create a cluster. IDCAMS is a utility, developed by IBM, for access method services. It is used to primarily define VSAM datasets.

Defining a Cluster

The following syntax shows the main parameters which are grouped under **Define Cluster, Data** and **Index**.

```
.DEFINE CLUSTER (NAME(vsam-file-name)       -
BLOCKS(number)                             -
VOLUMES(volume-serial)                     -
[INDEXED / NONINDEXED / NUMBERED / LINEAR] -
RECSZ(average maximum)                     -
[FREESPACE(CI-Percentage,CA-Percentage)]   -
CISZ(number)                               -
[KEYS(length offset)]                      -
[READPW(password)]                         -
[FOR(days)|TO(date)]                       -
[UPDATEPW(password)]                       -
[REUSE / NOREUSE] )                        -
DATA                                       -
   (NAME(vsam-file-name.data))             -
INDEX                                      -
   (NAME(vsam-file-name.index))            -
CATALOG(catalog-name[/password]))
```

Parameters at the CLUSTER level apply to the entire cluster. Parameters at the DATA or INDEX level apply only to the data or index component.

We will discuss each parameter in detail in the following table –

Sr.No	Parameters with Description
1	**DEFINE CLUSTER** Define Cluster command is used to define a cluster and specify parameter attributes for the cluster and its components.
2	**NAME** NAME specifies the name of VSAM file for which we are defining the cluster.
3	**BLOCKS** Blocks specifies the number of blocks assigned for the cluster.
4	**VOLUMES** Volumes specifies one or more volumes that will contain the cluster or component.
5	**INDEXED / NONINDEXED / NUMBERED / LINEAR** This parameter can take three values INDEXED, NONINDEXED or NUMBERED depending upon the type of dataset we are creating. For key-sequenced$KSDS$ files INDEXED option is used. For entry-sequenced$ESDS$ files the NONINDEXED option is used. For relative-record$RRDS$ files the NUMBERED option is required. For LinearLDS files the LINEAR option is required. The default value of this parameter is INDEXED. We will discuss more about KSDS, ESDS, RRDS and LDS in coming modules.
6	**RECSZ** Record Size parameter has two values which are Average and Maximum record size. The Average specifies the average length of the logical records in the file and the Maximum denotes the length of the records.

7	**FREESPACE** Freespace specifies the percentage of free space to reserve for the control intervals CI and control areas CA of the data component. The default value of this parameter is zero percentage.
8	**CISZ** CISZ is known as Control interval size. It specifies the size of control intervals.
9	**KEYS** Keys parameter is defined only in key-sequenced $KSDS$ files. It specifies the length and offset of primary key from first column. The range of value of this parameter are from 1 to 255 bytes.
10	**READPW** Value in READPW parameter specifies the password of read level.
11	**FOR/TO** The value of this parameter specifies the amount of time in terms of date and days for retaining the file. The default value for this parameter is zero days.
12	**UPDATEPW** Value in UPDATEPW parameter specifies the password of update level.
13	**REUSE / NOREUSE** REUSE parameter allows clusters to be defined that may be reset to empty status without deleting and re-defining them.
14	**DATA - NAME** The DATA part of the cluster contains the dataset name which contains the actual data of the file.

15	**INDEX-NAME** The INDEX part of the cluster contains the primary key and the memory pointer for the corresponding record in the data part. It is defined when a Key Sequenced cluster is used.
16	**CATALOG** Catalog parameter denotes the catalog under which the file will be defined. We will discuss about catalog separately in upcoming modules.

Example

Following is a basic example to show how to define a cluster in JCL –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DEFINE CLUSTER (NAME(MY.VSAM.KSDSFILE)  -
   INDEXED                                 -
   RECSZ(80 80)                            -
   TRACKS(1,1)                             -
   KEYS(5  0)                              -
   CISZ(4096)                              -
   FREESPACE(3 3) )                        -
   DATA (NAME(MY.VSAM.KSDSFILE.DATA))      -
   INDEX (NAME(MY.VSAM.KSDSFILE.INDEX))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create MY.VSAM.KSDSFILE VSAM file.

Deleting a Cluster

To delete a VSAM file, the VSAM cluster needs to be deleted using IDCAMS utility. DELETE command removes the entry of the VSAM cluster from the catalog and optionally removes the file, thereby freeing up the space occupied by the object. If the VSAM data set is not expired then it will not be deleted. To delete such types of datasets use PURGE option.

```
DELETE data-set-name CLUSTER
[ERASE / NOERASE]
[FORCE / NOFORCE]
[PURGE / NOPURGE]
[SCRATCH / NOSCRATCH]
```

Above syntax shows the parameters which we can use with Delete statement. We will discuss each of them in detail in following table –

Sr.No	Parameters with Description
1	**ERASE / NOERASE** ERASE option is specified to override the ERASE attribute specified for the object in the catalog. NOERASE option is taken by default.
2	**FORCE / NOFORCE** FORCE option is specified to delete the SPACE and USERCATALOG even if they are not empty. NOFORCE option is taken by default.
3	**PURGE / NOPURGE** PURGE option is used to delete the VSAM dataset if dataset has not expired. NOPURGE option is taken by default.
4	**SCRATCH / NOSCRATCH** SCRATCH option is specified to remove the associated entry for the object from the Volume Table of Contents. It is mainly used for non-vsam datasets like GDGs. NOSCRATCH option is taken by default.

Example

Following is a basic example to show how to delete a cluster in JCL –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEPNAME EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
   DELETE MY.VSAM.KSDSFILE CLUSTER
        PURGE
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will delete MY.VSAM.KSDSFILE VSAM file.

VSAM - ESDS

ESDS is known as Entry Sequenced Data Set. An entry-sequenced data set behaves like sequential file organization with some more features included. We can access the records directly and for safety purpose we can use passwords also. We must code **NONINDEXED** inside the DEFINE CLUSTER command for ESDS datasets. Following are the key features of ESDS –

- Records in ESDS cluster are stored in the order in which they were inserted into the dataset.
- Records are referenced by physical address which is known as **Relative Byte Address** *RBA*. Suppose if in an ESDS dataset, we have 80 byte records, the RBA of first record will be 0, RBA for second record will be 80, for third record it will be 160 and so on.
- Records can be accessed sequentially by RBA which is known as **addressed access.**
- Records are held in the order in which they were inserted. New records are inserted at the end.
- Deletion of records is not possible in ESDS dataset. But they can be marked as inactive.
- Records in ESDS dataset can be of fixed length or variable length.
- ESDS is non-indexed. Keys are not present in ESDS dataset, so it may contain duplicate records.
- ESDS can be used in COBOL programs like any other file. We will specify the file name in JCL and we can use the ESDS file for processing inside program. In COBOL program specify file organization as **Sequential** and access mode as **Sequential** with ESDS dataset.

Defining ESDS cluster

The following syntax shows which parameters we can use while creating ESDS cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

```
DEFINE CLUSTER (NAME(esds-file-name)     -
BLOCKS(number)                           -
VOLUMES(volume-serial)                   -
NONINDEXED                               -
RECSZ(average maximum)                   -
[FREESPACE(CI-Percentage,CA-Percentage)] -
CISZ(number)                             -
[READPW(password)]                       -
[FOR(days)|TO(date)]                     -
[UPDATEPW(password)]                     -
[REUSE / NOREUSE])                       -
DATA                                     -
   (NAME(esds-file-name.data))
```

Example

Following example shows how to create an ESDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
   DEFINE CLUSTER (NAME(MY.VSAM.ESDSFILE)  -
   NONINDEXED                              -
   RECSZ(80 80)                            -
   TRACKS(1,1)                          -
   CISZ(4096)                           -
   FREESPACE(3 3) )                     -
   DATA (NAME(MY.VSAM.ESDSFILE.DATA))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create MY.VSAM.ESDSFILE VSAM file.

Deleting ESDS Cluster

ESDS cluster is deleted using IDCAMS utility. DELETE command removes the entry of the VSAM cluster from the catalog and optionally removes the file, thereby freeing up the space occupied by the object.

```
DELETE data-set-name CLUSTER
[ERASE / NOERASE]
[FORCE / NOFORCE]
[PURGE / NOPURGE]
[SCRATCH / NOSCRATCH]
```

Above syntax shows which parameters we can use while deleting ESDS cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

Example

Following example shows how to delete an ESDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEPNAME EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DELETE MY.VSAM.ESDSFILE CLUSTER
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will delete MY.VSAM.ESDSFILE VSAM Cluster.

VSAM - KSDS

KSDS is known as Key Sequenced Data Set. A key-sequenced data set $KSDS$ is more complex than ESDS and RRDS but is more useful and versatile. We must code **INDEXED** inside the DEFINE CLUSTER command for KSDS datasets. KSDS cluster consists of following two components –

- **Index** – The index component of the KSDS cluster contains the list of key values for the records in the cluster with pointers to the corresponding records in the data component. Index component refers the physical address of a KSDS record. This relates the key of each record to the record's relative location in the data set. When a record is added or deleted, this index is updated accordingly.

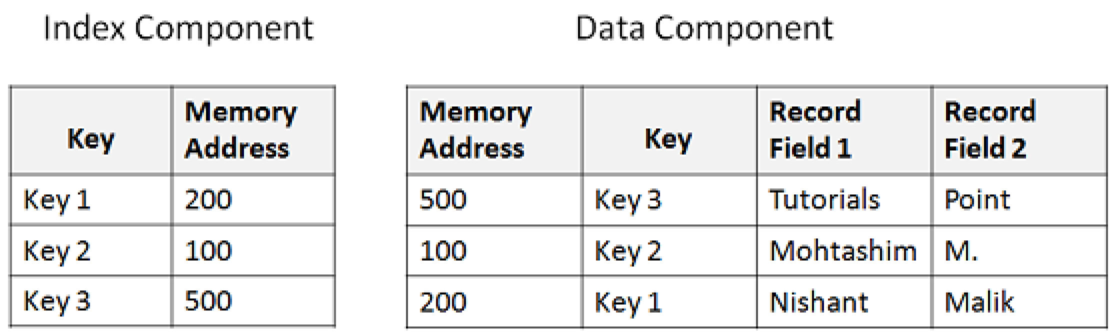
- **Data** – The data component of the KSDS cluster contains the actual data. Each record in the data component of a KSDS cluster contains a key field with same number of characters and occur in the same relative position in each record.

Following are the key features of KSDS –

- Records within KSDS data set are always kept sorted by key-field. Records are stored in ascending, collating sequence by key.
- Records can be accessed sequentially and direct access is also possible.
- Records are identified using a key. The key of each record is a field in a predefined position within the record. Each key must be unique in KSDS dataset. So duplication of records is not possible.
- When new records are inserted, the logical order of the records depends on the collating sequence of the key field.
- Records in KSDS dataset can be of fixed length or variable length.
- KSDS can be used in **COBOL** programs like any other file. We will specify the file name in JCL and we can use the KSDS file for processing inside program. In COBOL program specify file organization as **Indexed** and you can use any access mode $Sequential, RandomorDynamic$ with KSDS dataset.

KSDS File Structure

In order to search for a particular record, we give a unique key value. Key value is searched in the index component. Once key is found the corresponding memory address which refers to the data component is retrieved. From the memory address we can fetch the actual data which is stored in the data component. Following example shows the basic structure of index and data file –



Index Component

Key	Memory Address
Key 1	200
Key 2	100
Key 3	500

Data Component

Memory Address	Key	Record Field 1	Record Field 2
500	Key 3	Tutorials	Point
100	Key 2	Mohtashim	M.
200	Key 1	Nishant	Malik

Defining KSDS Cluster

The following syntax shows which parameters we can use while creating KSDS cluster.

The parameter description remains the same as mentioned in VSAM - Cluster module.

```
DEFINE CLUSTER (NAME(ksds-file-name)     -
BLOCKS(number)                           -
VOLUMES(volume-serial)                   -
INDEXED                                  -
KEYS(length offset)                      -
RECSZ(average maximum)                   -
[FREESPACE(CI-Percentage,CA-Percentage)] -
CISZ(number)                             -
[READPW(password)]                       -
[FOR(days)|TO(date)]                     -
[UPDATEPW(password)]                     -
[REUSE / NOREUSE])                       -
```

```
DATA                                    -
   (NAME(ksds-file-name.data))          -
INDEX                                   -
   (NAME(ksds-file-name.index))
```

Example

Following example shows how to create an KSDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DEFINE CLUSTER (NAME(MY.VSAM.KSDSFILE)  -
   INDEXED                                 -
   KEYS(6 1)                               -
   RECSZ(80 80)                            -
   TRACKS(1,1)                             -
   CISZ(4096)                              -
   FREESPACE(3 3) )                        -
   DATA (NAME(MY.VSAM.KSDSFILE.DATA))      -
   INDEX (NAME(MY.VSAM.KSDSFILE.INDEX))    -
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create MY.VSAM.KSDSFILE VSAM file.

Deleting KSDS Cluster

KSDS cluster is deleted using IDCAMS utility. DELETE command removes the entry of the VSAM cluster from the catalog and optionally removes the file, thereby freeing up the space occupied by the object.

```
DELETE data-set-name CLUSTER
[ERASE / NOERASE]
[FORCE / NOFORCE]
[PURGE / NOPURGE]
[SCRATCH / NOSCRATCH]
```

Above syntax shows which parameters we can use while deleting KSDS cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

Example

Following example shows how to delete an KSDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEPNAME EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DELETE MY.VSAM.KSDSFILE CLUSTER
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will delete MY.VSAM.KSDSFILE VSAM Cluster.

VSAM - RRDS

RRDS is known as Relative Record Data Set. RRDS cluster is similar to an ESDS cluster. The only difference is that RRDS records are accessed by **Relative Record Number** RRN, we must code **NUMBERED** inside the DEFINE CLUSTER command. Following are the key features of RRDS –

- A Relative record dataset has records that are identified by the **Relative Record Number** RRN, which is the sequence number relative to the first record.
- RRDS allows access of records by number like record 1, record 2, and so on. This provides random access and assumes the application program has a way to get the desired record numbers.
- The records in an RRDS dataset can be accessed sequentially, in relative record number order, or directly, by supplying the relative record number of the desired record.
- The records in a RRDS dataset are stored in fixed length slots. Each record is referenced by the number of its slot, number can vary from 1 to the maximum number of records in the dataset.
- Records in a RRDS can be written by inserting new record into an empty slot.
- Records can be deleted from an RRDS cluster, thereby leaving an empty slot.
- Applications which use fixed-length records or a record number with contextual meaning that can use RRDS datasets.
- RRDS can be used in **COBOL** programs like any other file. We will specify the file name in JCL and we can use the KSDS file for processing inside program. In COBOL program specify file organization as **RELATIVE** and you can use any access mode $Sequential, RandomorDynamic$ with RRDS dataset.

RRDS File Structure

Space is divided into fixed length slots in RRDS file structure. A slot can be either completely vacant or completely full. Thus, new records can be added to empty slots and existing records can be deleted from slots which are filled. We can access any record directly by giving Relative Record Number. Following example shows the basic structure of data file –

Data Component

Relative Record Number	Record Field 1	Record Field 2
1	Tutorial	Point
2	Mohtashim	M.
3	Nishant	Malik

Defining RRDS Cluster

The following syntax shows which parameters we can use while creating RRDS cluster.

The parameter description remains the same as mentioned in VSAM - Cluster module.

```
DEFINE CLUSTER (NAME(rrds-file-name)     -
BLOCKS(number)                           -
VOLUMES(volume-serial)                   -
NUMBERED                                 -
RECSZ(average maximum)                   -
[FREESPACE(CI-Percentage,CA-Percentage)] -
CISZ(number)                             -
[READPW(password)]                       -
[FOR(days)|TO(date)]                     -
[UPDATEPW(password)]                     -
[REUSE / NOREUSE])                       -
DATA                                     -
   (NAME(rrds-file-name.data))
```

Example

Following example shows how to create an RRDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DEFINE CLUSTER (NAME(MY.VSAM.RRDSFILE)  -
   NUMBERED                                -
   RECSZ(80 80)                            -
   TRACKS(1,1)                             -
   REUSE                                   -
   FREESPACE(3 3) )                        -
   DATA (NAME(MY.VSAM.RRDSFILE.DATA))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create MY.VSAM.RRDSFILE VSAM file.

Deleting RRDS Cluster

RRDS cluster is deleted using IDCAMS utility. DELETE command removes the entry of the VSAM cluster from the catalog and optionally removes the file, thereby freeing up the space occupied by the object.

```
DELETE data-set-name CLUSTER
[ERASE / NOERASE]
[FORCE / NOFORCE]
[PURGE / NOPURGE]
[SCRATCH / NOSCRATCH]
```

Above syntax shows which parameters we can use while deleting RRDS cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

Example

Following example shows how to delete an RRDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEPNAME EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DELETE MY.VSAM.RRDSFILE CLUSTER
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will delete MY.VSAM.RRDSFILE VSAM Cluster.

VSAM - LDS

LDS is known as Linear Data Set. Linear dataset is the only form of byte-stream dataset which is used in used in traditional operating system files. Linear datasets are rarely used. Following are the key features of LDS –

- Linear datasets does not contain RDF's and CIDF's as it does not have any control information embedded in its CI.
- Data that can be accessed as byte-addressable strings in virtual storage in Linear datasets.
- Linear datasets has a control interval size of 4KBytes.
- LDS is a kind of non-vsam file with some VSAM facilities like use of IDCAMS and VSAM specific information in the catalog.
- DB2 is currently the biggest user of Linear Data Sets.
- IDCAMS is used to define an LDS but it is accessed using a Data-In-Virtual *DIV* macro.
- Linear dataset does not have concepts of records. All LDS bytes are data bytes.

Defining LDS cluster

The following syntax shows which parameters we can use while creating LDS cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

```
DEFINE CLUSTER (NAME(lds-file-name)      -
BLOCKS(number)                           -
VOLUMES(volume-serial)                   -
LINEAR                                   -
CISZ(number)                             -
[READPW(password)]                       -
[FOR(days)|TO(date)]                     -
[UPDATEPW(password)]                     -
[REUSE / NOREUSE])                       -
DATA                                     -
   (NAME(lds-file-name.data))
```

Example

Following example shows how to create an LDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DEFINE CLUSTER (NAME(MY.VSAM.LDSFILE)   -
   LINEAR                                  -
   TRACKS(1,1)                             -
   CISZ(4096) )                            -
   DATA (NAME(MY.VSAM.LDSFILE.DATA))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create MY.VSAM.LDSFILE VSAM file.

Deleting LDS Cluster

LDS cluster is deleted using IDCAMS utility. DELETE command removes the entry of the VSAM cluster from the catalog and optionally removes the file, thereby freeing up the space occupied by the object.

```
DELETE data-set-name CLUSTER
[ERASE / NOERASE]
[FORCE / NOFORCE]
[PURGE / NOPURGE]
[SCRATCH / NOSCRATCH]
```

Above syntax shows which parameters we can use while deleting LDS cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

Example

Following example shows how to delete an LDS cluster in JCL using IDCAMS utility –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEPNAME EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DELETE MY.VSAM.LDSFILE CLUSTER
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will delete MY.VSAM.LDSFILE VSAM Cluster.

VSAM - COMMANDS

VSAM commands are used to perform certain operations on VSAM datasets. Following are the most useful VSAM commands –

- Alter
- Repro
- Listcat

- Examine
- Verify

Alter

ALTER command is used to modify VSAM file attributes. We can change the attributes of VSAM file which we have mentioned in VSAM Cluster definition. Following is the syntax to change the attributes –

```
ALTER  file-cluster-name [password]
   [ADDVOLUMES(volume-serial)]
   [BUFFERSPACE(size)]
   [EMPTY / NOEMPTY]
   [ERASE / NOERASE]
   [FREESPACE(CI-percentage CA-percentage)]
   [KEYS(length offset)]
   [NEWNAME(new-name)]
   [RECORDSIZE(average maximum)]
   [REMOVEVOLUMES(volume-serial)]
   [SCRATCH / NOSCRATCH]
   [TO(date) / FOR(days)]
   [UPGRADE / NOUPGRADE]
   [CATALOG(catalog-name [password]]
```

Above syntax shows which parameters we can alter in an existing VSAM cluster. The parameter description remains the same as mentioned in VSAM - Cluster module.

Example

Following example shows how to use ALTER command to increase Freespace, to add more volumes and to Alter Keys –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN DD  *
   ALTER  MY.VSAM.KSDSFILE
   [ADDVOLUMES(2)]
   [FREESPACE(6 6)]
   [KEYS(10 2)]
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will alter the Freespace, Volumes and Keys.

Repro

REPRO command is used to load data into VSAM dataset. It is also used to copy data from one VSAM data set to another. We can use this command to copy data from sequential file to VSAM file. IDCAMS utility uses REPRO command to load the datasets.

```
REPRO INFILE(in-ddname)
   OUTFILE(out-ddname)
```

In the above syntax, the in-ddname is DD name for the Input Dataset which is having records. The out-ddname is the DD name for the Output Dataset, where the input datasets records will be copied.

Example

Following example shows how to copy records from one dataset to another VSAM dataset –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//IN  DD DSN = MY.VSAM.KSDSFILE,DISP = SHR
//OUT DD DSN = MY.VSAM1.KSDSFILE,DISP = SHR
//SYSPRINT DD  SYSOUT = *
//SYSIN DD  *
   REPRO INFILE(IN)
      OUTFILE(OUT)
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will copy all the records from MY.VSAM.KSDSFILE to MY.VSAM1.KSDSFILE VSAM file.

Listcat

LISTCAT command is used to get the catalog details of a VSAM dataset. Listcat command provides following information about VSAM datasets –

- SMS Information
- RLS Information
- Volume Information
- Sphere Information
- Allocation Information
- Dataset Attributes

```
LISTCAT ENTRY(vsam-file-name) ALL
```

In the above syntax, vsam-file-name is the VSAM dataset name for which we need all the information. ALL keyword is specified to get all catalog details.

Example

Following example shows how to fetch all the details using Listcat command for a VSAM dataset –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN DD  *
   LISTCAT ENTRY(MY.VSAM.KSDSFILE)
   ALL
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will show all the catalog details about MY.VSAM.KSDSFILE dataset.

Examine

Examine command is used to check the structural integrity of a key-sequenced data set cluster. It checks for index and data components and if any problem is found, the error messages are sent spool. You can check any of the IDCxxxxx messages.

```
EXAMINE NAME(vsam-ksds-name) -
   INDEXTEST DATATEST -
   ERRORLIMIT(50)
```

In the above syntax, vsam-ksds-name is the VSAM dataset name for which we need to examine index and data part of VSAM cluster.

Example

Following example shows how to check whether Index and Data part of KSDS dataset are synchronized or not –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN DD  *
   EXAMINE NAME(MY.VSAM.KSDSFILE) -
   INDEXTEST DATATEST -
   ERRORLIMIT(50)
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will show all problems with the VSAM data set in one of the IDCxxxxx messages in spool.

Verify

Verify command is used to check and fix VSAM files which have not been closed properly after an error. The command adds correct End-Of-Data records to the file.

```
VERIFY DS(vsam-file-name)
```

In the above syntax, vsam-file-name is the VSAM dataset name for which we need to check the errors.

Example

Following example shows how to check and fix errors in VSAM dataset –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN DD  *
   VERIFY DS(MY.VSAM.KSDSFILE)
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will fix the errors in VSAM dataset.

VSAM - ALTERNATE INDEX

Alternate index are the additional index that are created for KSDS/ESDS datasets in addition to their primary index. An alternate index provides access to records by using more than one key. The key of alternate index can be a non-unique key, it can have duplicates.

Creation of Alternate Index

Following steps are used to create an Alternate Index –

- Define Alternate Index
- Define Path
- Building Index

Define Alternate Index

Alternate Index is defined using **DEFINE AIX** command.

```
DEFINE AIX                                -
(NAME(alternate-index-name)               -
RELATE(vsam-file-name)                    -
CISZ(number)                              -
FREESPACE(CI-Percentage,CA-Percentage)    -
KEYS(length offset)                       -
NONUNIQUEKEY / UNIQUEKEY                  -
UPGRADE / NOUPGRADE                       -
RECORDSIZE(average maximum))              -
DATA                                      -
   (NAME(vsam-file-name.data))            -
INDEX                                     -
   (NAME(vsam-file-name.index))
```

Above syntax shows the parameters which are used while defining Alternate Index. We have already discussed some parameters in Define Cluster Module and some of the new parameters are used in defining Alternate Index which we will discuss here –

Sr.No	Parameters with Description
1	**DEFINE AIX** Define AIX command is used to define Alternate Index and specify parameter attributes for its components.
2	**NAME**

	NAME specifies the name of Alternate Index.
3	**RELATE** RELATE specifies the name of the VSAM cluster for which the alternate index is created.
4	**NONUNIQUEKEY / UNIQUEKEY** UNIQUEKEY specifies that the alternate index is unique and NONUNIQUEKEY specifies that duplicates may exist.
5	**UPGRADE / NOUPGRADE** UPGRADE specifies that the alternate index should be modified if the base cluster is modified and NOUPGRADE specifies that the alternate indexes should be left alone if the base cluster is modified.

Example

Following is a basic example to show how to define an Alternate Index in JCL –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   DEFINE AIX (NAME(MY.VSAM.KSDSAIX)      -
   RELATE(MY.VSAM.KSDSFILE)               -
   CISZ(4096)                             -
   FREESPACE(20,20)                       -
   KEYS(20,7)                             -
   NONUNIQUEKEY                           -
   UPGRADE                                -
   RECORDSIZE(80,80))                     -
   DATA(NAME(MY.VSAM.KSDSAIX.DATA))       -
   INDEX(NAME(MY.VSAM.KSDSAIX.INDEX))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create MY.VSAM.KSDSAIX Alternate Index.

Define Path

Define Path is used to relate the alternate index to the base cluster. While defining path we specify the name of the path and the alternate index to which this path is related.

```
DEFINE PATH                        -
NAME(alternate-index-path-name)    -
PATHENTRY(alternate-index-name))
```

Above syntax has two parameters. NAME is used to specify the Alternate Index Path Name and PATHENTRY is used to specify Alternate Index Name.

Example

Following is a basic example to define Path in JCL –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
DEFINE PATH                          -
   NAME(MY.VSAM.KSDSAIX.PATH)    -
   PATHENTRY(MY.VSAM.KSDSAIX))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will create path between Alternate Index to the base cluster.

Building Index

BLDINDEX command is used to build the alternate index. BLDINDEX reads all the records in the VSAM indexed data set *orbasecluster* and extracts the data needed to build the alternate index.

```
BLDINDEX                           -
INDATASET(vsam-cluster-name)       -
OUTDATASET(alternate-index-name))
```

Above syntax has two parameters. INDATASET is used to specify the VSAM Cluster Name and OUTDATASET is used to specify Alternate Index Name.

Example

Following is a basic example to Build Index in JCL –

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS = A,MSGCLASS = C
//STEP1  EXEC PGM = IDCAMS
//SYSPRINT DD  SYSOUT = *
//SYSIN    DD  *
   BLDINDEX                            -
   INDATASET(MY.VSAM.KSDSFILE)         -
   OUTDATASET(MY.VSAM.KSDSAIX))
/*
```

If you will execute the above JCL on Mainframes server. It should execute with MAXCC = 0 and it will build the index.

VSAM - CATALOG

Catalog maintains the unit and volume where dataset resides. Catalog is used for retrieval of datasets. Non-VSAM datasets create a catalog entry by means of Disposition Parameter in JCL. VSAM datasets maintains its own catalog in form of KSDS cluster. In following image you can see the type of VSAM catalogs –

Catalog

Master Catalog

Master catalog is itself a file which monitors and manages operations of VSAM. Their is only one master catalog in any system which contains entries about system datasets and VSAM datasets. VSAM and Non-VSAM datasets may have entry in master catalog but this is not a good practice. The master catalog is created during system generation process and resides on system volume. Master catalog owns all VSAM resources in operating system. All files used in VSAM are controlled by master catalog. Master catalog is responsible for following operations –

- Password Authorization for files
- Enhancing the Security
- VSAM access for files
- Space Management of file
- Location of file
- Free Space available in file

When any of the above file attributes changes they are automatically updated in master catalog. Master catalog is defined using IDCAMS programs.

User Catalog

User catalog has same structure and concepts as the master catalog. It is present at next hierarchy level after master catalog. User catalog is not mandatory in the system but it is used to enhance security of VSAM system. Master catalog points to VSAM files but if User catalog is present then master catalog points to user catalog. User catalogs can be many in number as per the system requirement. In VSAM structure if master catalog is removed then it will not affect the user catalog. User catalog contain entries about application specific datasets. The information of user catalog is stored in master catalog.

Data Space

Data space is an area of the direct access storage device that is exclusively allocated for VSAM use. Data space must be created before creating VSAM clusters. The area occupied by the data space is recorded in the Volume Table of Contents $VTOC$, so that the space will not be available for allocation to any other use, either VSAM or non-VSAM. VTOC has entry of area occupied by space. VSAM creates a data space to hold the user catalog entries. VSAM takes control of this space and monitors and maintains this space as needed by VSAM files.

Unique Clusters

Unique Clusters consists of a separate data space which is utilized completely by the cluster created within it. Unique clusters are created out of unallocated space on direct access storage.

Sub-allocated Clusters

A sub-allocated VSAM file shares the VSAM space with other sub-allocated files. It specifies that file should be sub-allocated within existing VSAM space. Sub-allocation is used for easier management and control of VSAM spaces.

Non-VSAM datasets

Non-VSAM datasets resides on both tape and direct access storage. Non-VSAM datasets may have entries in both master catalog and user catalogs. The main function of cataloging non-VSAM datasets is to retain unit and volume serial information.

VSAM - FILE STATUS

While working with VSAM datasets you may encounter abends. Following are the common file status codes with their description which will help you to resolve the issues –

Code	Description
00	Operation completed successfully
02	Non-Unique Alternate Index duplicate key found
04	Invalid fixed length record
05	While performing OPEN File and file is not present
10	End of File encountered
14	Attempted to READ a relative record outside file boundary
20	Invalid Key for VSAM KSDS or RRDS
21	Sequence error while performing WRITE or changing key on REWRITE
22	Primary duplicate Key found
23	Record not found or File not found
24	Key outside boundary of file
30	Permanent I/O Error
34	Record outside file boundary
35	While performing OPEN File and file is not present
37	OPEN file with wrong mode
38	Tried to OPEN a Locked file
39	OPEN failed because of conflicting file attributes

41	Tried to OPEN a file that is already open
42	Tried to CLOSE a file that is not OPEN
43	Tried to REWRITE without READing a record first
44	Tried to REWRITE a record of a different length
46	Tried to READ beyond End-of-file
47	Tried to READ from a file that was not opened I-O or INPUT
48	Tried to WRITE to a file that was not opened I-O or OUTPUT
49	Tried to DELETE or REWRITE to a file that was not opened I-O
91	Password or authorization failed
92	Logic Error
93	Resources are not available
94	Sequential record unavailable or concurrent OPEN error
95	File Information invalid or incomplete
96	No DD statement for the file
97	OPEN successful and file integrity verified
98	File is Locked - OPEN failed
99	Record Locked - record access failed