ARCHIVER 6.1.2

The Systems Programmer's Toolkit manager

Abstract

This document and the program it describes may be distributed freely, to any interested party. No fees or charges are permitted, other than the recovery of reasonable distribution costs. Users are encouraged to submit suggestions for improvement to the author, via NASPA.

Rick Fochtman

Table of Contents

General

In my many and varied ramblings, I've encountered, and developed, many programs, subroutines, macros and clists that I've found useful in my work.

Because I was changing jobs at a rapid rate at that time, I wanted a mechanism that provided a certain amount of portability for this 'tool kit'.

Net result: the ARCHIVER program. I wanted to be able to package everything in a simple-to-manage form that would be essentially device-independant. I selected VSAM as my 'bridge' because that seems to be the direction of the IBM data management trend. My basic design criteria were these:

1. Ability to access partitioned and sequential datasets of nearly any format. I wasn't worried about spanned records, since they are unusable for most of the functions I needed. Just about anything else is acceptable to the ARCHIVER.
2. Where possible, utilize IBM utilities and mechanisms for the more mundane functions, such as backup and reorganization.
3. Provide a mechanism for multiple occurrences of a PDS member name, since source, clist and load module might be all the same name.
4. Provide a mechanism for multiple versions, since I might encounter compatibility problems in some routines.
5. Allow for historical and/or comment data, to help me remember the function of each archived item.
6. As far as possible, remain independent of the MVS software level differences. For that reason, I chose VSAM but didn't use some of the more advanced functions of VSAM.

All these criteria were met in the initial version, with a few minor limitations. For example, overlay load modules were improperly processed since the note list wasn't updated during reload. During the ensuing evolution, the ARCHIVER has changed radically, accepting new functions and polishing some existing functions. Soon, I hope to be able to ARCHIVE any dataset that can be processed using either QSAM or BPAM.

This document you are reading corresponds to Version 5.0 of the ARCHIVER.

Although functions of older versions are still present, they may function differently in Version 5.0. Let the user beware.

Changes from Previous Versions:

Evolution of the ARCHIVER, with prompting from Sam Golob, has resulted in a new feature, the ALIAS functions. The VSAM ARCHIVE may now contain any number of aliases for a given item, allowing the same item to be referenced by several sets of qualifiers. In previous versions, if a user wished to include a macro, for example, under several sets of qualifications, he was forced to include several copies of the macro. No more. The ADDALIAS command allows the user to assign aliases in virtually any pattern he desires.

The DELETE function has been EXTENSIVELY reworked, both to handle the new aliases and for general cleanup.

The EXPORT and IMPORT functions of ARCHIVER now use a DD name or DSNAME operand that is unique to EXPORT/IMPORT. While this may seem trivial to someusers, it was somewhat confusing to use the LOADT and LISTT functions.

Two new functions allow a special list of the number of items in each category, where a category is a set of GROUP/SUBGROUP, TYPE and VERSION qualifiers that match.

The latest and largest enhancement is a COMPARE function. This command allows a user to compare items, of sets of items, from the single archive cluster, or optionally, compare like-named items from two different archive clusters, all under user control. Eventually, ARCHIVER will automatically delete duplicates and assign aliases, further reducing the space requirements for the archive cluster.

The ARCHIVE dataset

The ARCHIVE dataset is a key-sequenced VSAM cluster, containing three distinct key ranges. The actual key of the record contains a record type, the qualifier information and a sequence number, in that order. The record type is the mechanism that sets the key range. The record types are header, note-data and item data. While the note-data record is optional, all other types are required. Here's a short description of each type.

* HEADER RECORDS are the root of all archived items. The header record has the required key, plus the RECFM, LRECL, BLKSIZE, DSORG and DEVTYPE of the item's source dataset, as well as the time and date of insertion in the archive and the record count of the original member. Also, in version 4, all directory data, including PDS alias information, is stored in the header record. The size and number of directory records is such that, if stored separately, much space and time are wasted, to no good purpose. (Previous versions of the ARCHIVER used a separate record type for PDS directory data.)

* NOTE-DATA records, which may not be present, contain comment data supplied by the user for the item described by the record key. Multiple records may be present, with ascending sequence numbers to maintain the uniqueness of the key. As in the item data records, all the data is in a compressed format, for space-economy reasons.

* ITEM DATA records contain the actual logical records of the item, in compressed format. Previous versions of the ARCHIVER used a fairly primitive compression mechanism that only compressed out repetitive strings. This new improved version uses a variation of the Huffman Tree to do both compression of repetitive strings AND compaction of data at the bit level.

VSAM was selected as the access method for the Archive dataset for two very important reasons. First, because the user of VSAM has no great need to know the characteristics of the actual device. Second, standard IBM-type utility programs could be used for backup, space recovery, creating transportable copies, etc. While this may seem trivial, the problems involved in developing a completely new access mechanism just aren't justified by the potential savings.

For reference, here's a sample of the IDCAMS control statements required to allocate the archive cluster. Note that there is no provision for 'seed'ing the cluster, since the ARCHIVER will automatically deal with this chore.

```
DEFINE CLUSTER(NAME( your archive name ) -
RECORDSIZE(200 32000) -
FREESPACE(20 20) -
BUFFERSPACE(262144) - (note)
KEYS(49 0))
DATA(NAME( your archive name .DATA) -
CYLINDERS( primary secondary )) -
INDEX(NAME( your archive name .INDEX) -
```

```
CYLINDERS( primary secondary ))
```

Note: while this value may seem excessive, it greatly enhances the overall performance of the ARCHIVER.

Maximum record size should be at least 90 bytes larger than the longest load-module record to be unloaded. (control information)

REPLICATE/IMBED are performance options that may or may not enhance the performance of ARCHIVER, depending on usage patterns.

Because of the way VSAM compresses the record keys, a single cylinder of index is enough for about 10,000 archived items.

RECORD KEYS

Because of a design requirement to be able to store multiple PDS members without the necessity of renaming, and confusing, each item has four levels of qualification possible in the archive cluster. These qualifiers are named MEMBER, GROUP, SUBGROUP and TYPE within the ARCHIVER program.

Externally, they can be thought of in those terms or in any other 'style' that may be convenient. The original intent was to be able to group together items that were related for ease in remembering everything. As an example, if you wished to unload the entire CBT mods tape to the archive, then the GROUP could be CBTTAP, the subgroup could be the original CBT tape file number, i.e. FILE147 and the type could be indicative of the programming language, i.e. BAL. If you choose to specify a version number to the ARCHIVER, that could be the CBT tape version number, e.g. 311.

The version number is processed in a different fashion from the other key values, since it is maintained in DESCENDING sequence. This is done by using the NEGATIVE of the version number. Please note however that a negative version number is not properly processed by the ARCHIVER's input scan, so results may be unpredictable if this is attempted. Version numbers are considered qualifier data, thus may be part of the user's item selection criteria for processing. In all cases, the version number is STRICTLY OPTIONAL and in some cases its use should be avoided.

ALIAS PROCESSING

Version 5.0 of the ARCHIVER recognizes, and processes, two distinct 'flavors' of aliases: BPAM aliases, such as we're all accustomed to using, and ARCHIVER aliases, a completely new feature in this version.

BPAM aliases (also known as PDS aliases), are saved in the header record for each item that is unloaded. Naturally, sequential datasets have no aliases. During reload processing, these aliases, if present, may be restored or ignored at the user's discretion.

As yet, there is no mechanism within the ARCHIVER to list the PDS directory data for each archived item; that's coming but it's going to be some time yet. There are too many other things that the ARCHIVER needs.

ARCHIVER aliases, on the other hand, are completely under user control. The user may automatically assign ARCHIVER aliases during unload, add, delete or alter aliases and list, or not list, these aliases. If an item is to be deleted, the actual data can be reassigned to a different set of qualifier data and retained, or discarded completely with all its aliases, entirely at the user's discretion. In situations where multiple archive datasets are being merged, this can result in a massive space saving.

ARCHIVER JCL

Job control language for the ARCHIVER is quite simple and straightforward. Required DD statements are SYSPRINT and SYSIN.

You can use an alternate DDNAME of SYSPRINT with PARM='SYSPRINT=ddname' in the EXEC statement of ARCHIVER and You can use an alternate DDNAME of SYSIN, with PARM='SYSIN=ddname' in the EXEC statement of ARCHIVER. If you want to override both ddnames, code PARM='SYSIN=ddname1,SYSPRINT=ddname2' Period.

Additional DD statements MAY be required for some functions but the Dynamic Allocation facilities within the ARCHIVER program serve to greatly reduce JCL requirements over previous versions. Any CATALOGED dataset may be allocated dynamically, without exception.

The SYSPRINT dataset is, obviously, the report and listing dataset used by all ARCHIVER functions. Record format is FBA and logical record length is set to 133. If blocksize is not supplied, the default is 133.

Again obviously, the SYSIN dataset is the control statement input dataset.

Record format is set to FB and logical record length is 80. Again, if a blocksize is not supplied, from the DD statement or a dataset label, the default value is 80.

General Control Statement Format

The ARCHIVER control statements all follow a fairly simple and familiar format, using all keyword= style operands. While the major operands are all of the keyword=value format, a few of the operands are of a multiple and positional format. I know that's probably hard to understand but when you examine the individual control statement formats, you'll see almost instantly what it means. Control statements are continued by ending a statement with a comma followed by a blank. The continuation may begin in any position, for any length, provided it does not extend past column 72.

The number of continuations permitted is unlimited, with the provision that a control statement may not contain more than 2048 characters total.

The general form of the ARCHIVER control statement is this:

```
verb keyword=value,...,keyword=value
```

Item selection is slightly more complicated because of the various types of qualifier data that may be used to affect selection. The ITEM= value is actually a list of either four or five positional values, corresponding to name, group, subgroup, type and version number, in that order. The first four qualifiers may be any character string, except a blank, and may be specific values or generic values. The version number must always be a numeric value in the range of 1-16777216.

Generic values are denoted in any of several ways.

If a qualifier is completely replaced by an asterisk, the ARCHIVER assumes that ANY value, of any length, is a valid match for this qualifier. Thus the parameter 'ITEM=(*,CBT321,FILE241,OBJ)' denotes any item whose group, subgroup and type are CBT321,FILE241 and OBJ, respectively.

If a qualifier is ended with an asterisk, all characters up to the asterisk must match for comparisons within the ARCHIVER. Any characters in the position of the asterisk, or after that position, are

ignored. In the example above, if the asterisk is replaced with 'RMF*', only members whose names start with RMF are considered to match.

If a qualifier contains one or more percent signs ('%'), these are considered to match any character in that position. Again using the example above, if we replace the asterist first qualifier value with 'RMF%%%01', only those values that have 'RMF' as the first three positions, and have '01' in positions 7-8 and are exactly eight characters long are considered to match.

The characters in positions 4-6 are ignored.

As you can see, the ability to use generics is a very powerful feature. Comment statements may be imbedded anywhere in the input stream. A comment is denoted by an asterisk ('*') in column one and will be printed as-is and ignored.

The SET Control Statement

The SET control statement allows the user to set default DDNAMES or DSNAMES for subsequent processing.

Format:

```
SET (EXPTPDD=ddname | EXPTPDSN=non-vsam dsname)

SET (VSAM1DD=ddname | VSAM1DSN=cluster name)

SET (VSAM2DD=ddname | VSAM2DSN=cluster name)

SET (DDN=ddname | DSN=non-VSAM dsname)
```

The operands for the SET command are DD names or DSNAMEs that will remain in effect for subsequent processing, unless one or more operands from the SET command appear on subsequent ARCHIVER control statements. Since some of these values are required for further processing, each control statement description will contain a note describing the SET operands that are required.

Because the operands using dsnames force DYNAMIC ALLOCATION (SVC-99) of the datasets named, all datasets named in the SET control statement MUST be CATALOGED DATASETS.

Performance note: because I use the ARCHIVER extensively, I've learned that using a DD statement to describe the VSAM cluster and inserting the BUFND and BUFNI subparameters of the AMP parameter can vastly improve performance. I generally use about 16 buffers for each component. Also, if the VSAM cluster is defined with GENEROUS bufferspace, it help performance. (I have defined my ARCHIVE clusters with 256k of bufferspace.)

The ADDALIAS Command

The ADDALIAS command allows the user to assign ARCHIVER-type aliases to items or groups of items in the Archive dataset.

General Format:

```
ADDALIAS ITEM=(...),ALIAS=(...)
```

The ITEM= values are as described previously in this manual, including theusage of generic qualifier data.

The ALIAS= operand follows the same format as the ITEM= operand but is processed differently. If a qualifier level in the ALIAS= operand is replaced completely by an asterisk, the corresponding qualifier from the ITEM= operand is copied. For example, suppose I wish to give the alias

CBT324,FILE247,DATA to all items currently qualified as CBT321,FILE247,DATA, my control statement would look like this:

```
ADDALIAS ITEM=(*,CBT321,FILE247,DATA),ALIAS=(*,CBT324,*,*)
```

The SET operand that must be in effect is: VSAM1DD or VSAM1DSN

The ADDNOTE Command

The ADDNOTE statement causes the ARCHIVER to insert NOTE-DATA into the members, or groups of members, named in the control statement and present in the VSAM1 archive dataset. Since this is a relatively complex function to understand, a sample job stream is provided to illustrate this function.

Format:

```
ADDNOTE ITEM=(...),KEY=value
```

The ADDNOTE control statement takes some practice to understand fully. It uses the non-VSAM dataset data, which must be card-image data, as the comments or NOTES to be added to the selected item. The KEY= value refers to a character string of 1-8 characters in position 73-80 of the NOTES records.

What happens is this: the non-VSAM dataset is read sequentially and each card image that has the key value in position 73-80 is added to the item as note data, minus the last 8 columns. For the next item, the non-VSAM dataset is closed and reopened and re-read, etc. The sample JCL streams distributed with the ARCHIVER illustrate this function and are perhaps easier to follow than my futile attempts here.

One thing to be aware of: all NOTE data is CUMULATIVE. If you wish to replace the note data associated with a particular item, you must first delete any pre-existing data, using the delete function.

The ADDNOTE function REQUIRES that the non-VSAM dataset support the REREAD option of CLOSE. While JES2 SYSIN seems to allow this, I just don't know whether JES3 will permit this function.

Required SET operand that must be in effect:

VSAM1DD= or VSAM1DSN= (denoting the Archive dataset)

DDN= or DSN= (denoting the file of input NOTE data)

The ALTALIAS Command

The ALTALIAS command allow the user to reassign an Archive alias to a different 'real' member. For example, if the alias XYZ,CBT321,FILE1,DATA is an alias for XYZ,CBT249,FILE1,DATA, perhaps you would like it to be an alias to XYZ,CBT998,FILE1,DATA instead. The ALTALIAS function will make this change.

General format:

```
ALTALIAS ITEM=(...),ALIAS=(...)
```

The ITEM= operand is processed as perviously described, including the processing of generics.

The ALIAS= operand is processed in its own special way for this function.

Each qualifier that is replaced by an asterisk remains unchanged. Any qualifier that is NOT replaced by an asterisk is used as a replacement.

Thus, to make the alias change described above, the control statement would be:

```
ALTALIAS ITEM=(XYZ,CBT249,FILE1,DATA),ALIAS=(*,CBT998,*,*)
```

The required SET operands that must be in effect for ALTALIAS are:

VSAM1DD= or VSAM1DSN= (the Archive dataset)

The ALTER Command

The ALTER control statement allows the user to change the qualifier data on any item, or group of items, in the ARCHIVE cluster. The function is carried out by rewriting the item(s) with new keys according to the qualifier data provided in the control statement.

General Format:

```
ALTER ITEM=(...),NEW=(...)
```

The NEW= operand is of the same format as the ITEM= operand and will be examined using the same mechanism. However, in the NEW= operand, generic processing is somewhat different. The generic characters '*' and '%' are not treated as generics but as normal characters. The only exception is the case where a complete qualifier is replaced by '*', which signals the ARCHIVER that this qualifier is to remain unchanged.

In the ITEM= operand, normal generic processing is in effect and will be honored. (Previous version of the ARCHIVER referred to this mechanism as RESPECIFY and the RESPECIFY command is considered an alias for ALTER in this version.)

Be aware that this function will cause the ARCHIVE to "grow", with splits, extents, etc. It's probably a good idea to reorg the ARCHIVE cluster after extensive use of ALTER.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the Archive dataset)

The CLEAN Command

The CLEAN command causes the ARCHIVER to check all the Archive aliases and remove those which might have had their 'real' item deleted by previous processing errors. Since this is a 'remedial' command, it should never be needed. Murphy's Law being what it is, the function was added anyway.

General format:

```
CLEAN
```

The CLEAN command has no required operands and will only permit the required SET operands to be included.

The required SET operands that must be in effect are:
VSAM1DD= or VSAM1DSN= (denoting the Archive dataset)

The COMPARE Command

The COMPARE statement causes the Archiver to compare two items, or groups of items, and message the results of that comparison. In this initial version, messages are very simple and short: either the items match or they don't match in some significant way. Fields compared include the original DCB attributes and record counts, item data and NOTE data. (If the NOTE data doesn't

match, this is not considered an unsuccessful compare, since the NOTE data doesn't participate in LOAD/LOADT functions.)

General Format:

```
COMPARE ITEM=(...),COMPTO=(....),SHRTLST=Y|N
```

All of the qualifier data for the ITEM keyword is optional and may be replaced by the generic value '*'. The format of the COMPTO operand is exactly the same as the ITEM operand, with one very important exception. In the COMPTO operand, if a qualifier is replaced by a '*', the compare function will search for an item with the SAME qualifier in that position as in the ITEM keyword. For example, consider this scenario:

```
COMPARE ITEM=(*,CBT325,*,*),COMPTO=(*,CBT324,*,*)
```

During the search of the VSAM1 dataset, an item is found with qualifiers of FILE001,CBT325,PDS84,PANEL. The compare function will attempt to locate, in the VSAM2 dataset, an item with the qualifiers of FILE001,CBT324,PDS84,PANEL for a comparison. If no such item is found, a message to that effect is produced and the next matching item from VSAM1 is located and compared to its matching VSAM2 item. Partial generics, using the '%' character as a place-holder are still honored, as are version numbers.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (input for ITEM= items)
VSAM2DD= or VSAM2DSN= (input for COMPTO items. may be the same dataset as VSAM1.)

The Archiver comparison checks the logical record length, record format, logical record count, LMOD indicators, not data in its entirety and item data in its entirety. The first failure to match terminates the comparison with appropriate messages. Also check, but only for information purposes, are unload times and dates.

The CONVERT Command

The CONVERT statement will cause the conversion from a previous version ARCHIVE cluster of all or selected data to the Version 5 ARCHIVE cluster.

Data compression/compaction mechanisms are converted as needed. This mechanism is provided for those users who have installed, and are using, previous version of the ARCHIVER program. Item selection is available and all generics are honored. No pre-existing item on the VSAM1 cluster is replaced. If a replacement would occur, the version number of the item to be added is incremented by one until insertion is possible.

General format:

```
CONVERT ITEM=(...)
```

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the Version 4 Archive dataset)
VSAM2DD= or VSAM2DSN= (denoting the previous version Archive dataset)

Please take careful note: VSAM clusters used and created by Archiver V4.0 DO NOT REQUIRE CONVERSION. The ARCHIVE format is unchanged from Version 4.

The COPY Command

The COPY statement causes the ARCHIVER to copy from the VSAM2 ARCHIVE cluster to the VSAM1 ARCHIVE cluster.

General format:

```
COPY ITEM=(...),REPLACE=Y|N,ALIASES=Y|N,CHKALIAS=Y|N
```

If the replace operand is supplied, it is honored. Otherwise, a duplicate item that is copied has the version number increased by one until the copy can succeed. The user should be aware: this operation can be very slow; large numbers of large items are to be avoided. You should also plan on a reorg of the VSAM1 cluster if you use this function much. If Archiver-type aliases are to be copied, the the ALIASES=Y keyword may be provided; and if Archiver-type aliases are eligible for selection, the CHKALIAS=Y keyword may be supplied as well.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the OUTPUT Archive dataset)
VSAM2DD= or VSAM2DSN= (denoting the INPUT Archive dataset)

The DELALIAS Command

The DELALIAs command works in either of two modes to delete aliases from the Archive dataset. It will either delete all aliases for a selected set of 'real' items, or delete all aliases having the qualifier data specified. The decision is based on the operand keyword supplied.

General Format:

```
DELALIAS ITEM=(...) - or - DELALIAS ALIAS=(...)
```

If the ITEM= operand is supplied, all aliases for any 'real' item that matches the qualifier data supplied will be deleted. If the ALIAS= operand is supplied, all ALIASES that match the qualifier data supplied will be deleted. In either case, the CLEAN processor is invoked after all alias deletion is complete.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the OUTPUT Archive dataset)

The DELETE Command

The DELETE function is just exactly what its name implies. The user may delete NOTE-DATA, back-level copies of items, entire groups of items or the entire ARCHIVE contents, depending on the choice of operands. Since an object that is deleted from the Archive cluster may not be recoverable, the DELETE function has NO DEFAULTS. The user must explicitly specify ALL options and keywords for the delete function.

General Format:

```
DELETE ITEM=(...),DATA=Y|N,NOTE=Y|N,BACK=Y|N,ALIASES=Y|N
```

Operands:

DATA=Y or DATA=N determines whether the actual item is to be deleted.

NOTE=Y or NOTE=N determines whether all note data should be deleted from the selected item(s). Use of the combination NOTE=N,DATA=Y will result in an error message and no action will be taken. Note data may not be held in the Archive without a 'real' item.

BACK=Y or BACK=N will determine whether 'back levels' are to be deleted. Thus, if the Archive contains multiple levels of an item, perhaps placed there as checkpoints in a development cycle, this operand will allow the deletion of these intermediate items. If you wish to delete ALL occurrances of a particular item, you must use the DELETE twice, once with BACK=Y and again with BACK=N.

ALIASES=Y or ALIASES=N has a little more meaning, and function, than the other operands. If ALIASES=Y is supplied, the item and all its aliases are deleted beyond recovery. If ALIASES=N is supplied and the item has aliases, the first alias is converted to a 'real' item and the data and note-data are altered to those qualifiers. Then the selected item is deleted. It's rather like having multiple copies of the item and notes for the item and just deleting one copy.

A cautionary note: the default operands are DATA=N,NOTE=N,ALIASES=N,BACK=N for good reason. Since deletes have the potential to cause permanent data loss, the user is required to supply ALL operands. Invalid combinations will result in error messages and no actions, rather than proceeding on possibly erroneous assumptions.

A further note: you may delete only the 'current' version of an item by specifying BACK=N. Each time you do this, the highest-numbered version will be deleted.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the OUTPUT Archive dataset)

The EXPORT Command

The EXPORT statement will cause the ARCHIVER to copy selected items from the ARCHIVE cluster to a sequential dataset in a special format that allows for a quick reload to another ARCHIVE cluster, or directly to non-VSAM datasets for use.

General Format:

```
EXPORT ITEM=(...)
```

Item selection is possible and generics are honored. The output dataset must be a QSAM-supported dataset and the user may specify only the BLKSIZE DCB parameter. The RECFM is VBS and the logical record length is fixed at 32760. If the user chooses not to supply a blocksize, the default is 32000.

Archive aliases are preserved during export processing and may be restored during IMPORT functions or ignored, at the user's discretion. The capability of examining the contents of this EXPORT dataset are provided through the LISTT command, described elsewhere.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the input Archive dataset)
EXPTPDD= or EXPTPDSN= (output sequential dataset)

The IMPORT Command

The IMPORT statement will cause the ARCHIVER to copy selected items from the dataset created by EXPORT into the Archive dataset, preserving or discarding Archive aliases at the user's discretion.

General Format:

```
IMPORT ITEM=(...),ALIASES=Y|N
```

Operands:

ALIASES=Y or ALIASES=N determines whether Archive aliases are to be restored. If the user specifies ALIASES=Y, all applicable aliases are restore as each item is restored. Please take careful note: the version number in the alias may be altered.

REPLACE=Y or REPLACE=N determines whether an item being restored should replace an pre-existing item with all matching qualifiers. If REPLACE=N is specified, the item being imported will be altered to form a new higher version, rather than replacing a pre-existing item.

Item selection is possible and generics are honored. The input dataset must be QSAM-supported and on DASD or TAPE.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (denoting the output Archive dataset)
EXPTPDD= or EXPTPDSN= (input sequential dataset)

The LFILE Command

New in this version of the Archiver is the concept of LOGICAL FILES. In this context, a LOGICAL FILE is any item or group of items that have a common GROUP, SUBGROUP, TYPE and VERSION NUMBER. In keeping with this idea, the program can list these logical files, giving item counts and total record counts. This allows the user to list, for example, all the files of a CBT tape that has been transferred into any archive, showing the number of members from each PDS and the number of logical records, very helpful for space estimates. (The SAMPLIB dataset contains an example of how I Archived the CBT325 tape.)

Command format:

```
LFILE ITEM=(.....)
```

While the ITEM= values are not changed for this command, the member name, the first field of the ITEM value, is ignored.

SET operands required:

VSAM1DD or VSAM1DSN is required.

The LFILET Command

In keeping with the idea of allowing as much as possible from the Archiver export dataset, the LFILET will perform the same function as the LFILE, the only difference being that the input is from the EXPTPDD/EXPTPDSN dataset, rather than the VSAM1DD/VSAM1DSN VSAM cluster.

Command format:

```
LFILET ITEM=(.....)
```

While the ITEM= values are not changed for this command, the member name, the first field of the ITEM value, is ignored.

SET operands required: EXPTPDD/EXPTPDSN is required.

The LIST Command

The LIST statement allows the user to list, directly from the VSAM1 ARCHIVE cluster, the contents of the ARCHIVE, in a variety of formats.

General Format:

```
LIST ITEM=(...),DATA=Y|N,NOTE=Y|N,HEX=Y|N,ALIAS=Y|N
```

DATA=Y or DATA=N determines whether the actual item data should be listed in its entirety.

HEX=Y or HEX=N determines whether the item data should be listed in a hexadecimal format, similar to a dump.

NOTE=Y or NOTE=N determines whether any NOTE data should be listed. Note data is always listed in character format, regardless of whether or not HEX=Y was specified.

ALIASES=Y or ALIASES=N determines whether Archiver aliases should be listed. The aliases are listed in a modified format, showing the qualifiers of the corresponding real item.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (input Archive dataset)

The LISTT Command

The LISTT function will list the contents of a dataset created by the EXPORT function of ARCHIVER. For most operands, refer to the LIST function. One notable exception: since the listing is of a non-VSAM dataset, the SET values in effect MUST include EXPTPDD or EXPTPDSN.

Since the input is an EXPORT of a VSAM Archive, the items listed will probably not be in perfect alphabetical order. All the Archive aliases for an item will be displayed immediately after the 'real' item name, since that's how they're exported.

The LOAD Command

The LOAD statement causes the ARCHIVER to copy from a VSAM ARCHIVE cluster to a non-VSAM dataset.

General Format:

```
LOAD ITEM=(...),REPLACE=Y|N,SEP=IEBUPDTE,ALIASES=Y|N,CHKALIAS=Y|N
```

The selected item(s) are copied from the ARCHIVE cluster to the non-VSAM dataset, reblocking and changing formats as requested, if possible. If a DCB format change is impossible, or if the record lengths are incompatible, an error message is produced to this effect and processing is terminated for the offending item.

REPLACE=Y|N determines whether pre-existing members in the output dataset will be replaced or not replaced during this LOAD.

ALIASES=Y|N determines whether any BPAM aliases for PDS members will be restored during the LOAD function.

CHKALIAS=Y|N determines whether Archive aliases will be considered during the LOAD function. If the value specified is Y (or YES), each alias will be checked against the criteria specified in the ITEM= parameter. The first time an alias matches this criteria, the item will be loaded. Each alias or non-alias that matches the 'real' item name after that point will be ignored.

Because of the way this is implemented, each 'real' member that is selected for loading will cause the ARCHIVER's storage requirements to grow by 48 bytes. Executing the LOAD function against a large number of Archive items may require significant increases in region size.

Please note: if the output dataset from the LOAD function is a NEW dataset, the DCB on the DD statement MUST specify the DSORG parameter, either PS or PO.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (input Archive dataset)
DD= or DSN= (output sequential or partitioned dataset)

The LOADT control statement

The LOADT statement causes the ARCHIVER to copy directly from a sequential dataset created by the ARCHIVER EXPORT function to a non-VSAM dataset.

All operands are exactly the same as for the LOAD statement, except that the input dataset (the EXPORTED archive) is designated by the EXPTPDD/EXPTPDSN operand of the SET command. This is a clarification from previous levels of the Archiver.

The UNLOAD Command

The UNLOAD statement causes the ARCHIVER to copy from a non-VSAM dataset into the ARCHIVE VSAM cluster described by the VSAM1DDN or VSAM1DSN parameter.

General Format:

```
UNLOAD ITEM=(...),ULMODS=Y|N,ALIASES=Y|N
```

All of the qualifiers must be specified except the mbr value, which may be generic if the non-VSAM dataset is a partitioned dataset. If a version number is supplied, any previously unloaded item with all qualifiers matching will be replaced. If no version number is supplied and a previously unloaded item exists with mbr, group, subgroup and type matching, a new version will be created. Version numbers should not be supplied, since it's easier to delete a bad version that to replace an overlaid item.

ULMODS= signals the ARCHIVER whether or not to treat any input data with a RECFM of U as LOAD MODULES. While this has little effect on non-load-module data, it will significantly enhance the processing of load modules during reload. Such things as tracking note lists, etc. The default is Y and the user would be well advised to leave this default in effect.

ALIASES= signals the ARCHIVER whether or not to create Archive aliases that correspond to the aliases that a PDS member might have in the dataset being unloaded. Here again, the default is Y and the user would be well advised to leave this default in effect.

The required SET operands that must be in effect are:

VSAM1DD= or VSAM1DSN= (output Archive dataset)
DD= or DSN= (input sequential or partitioned dataset)