PROLOGUE - The virtual lookaside facility (VLF) is a set of z/OS services that provide a high-performance alternate path method of retrieving named objects from DASD on behalf of many users. VLF is designed primarily to improve the response time for such applications.

* VLF uses data spaces to hold data objects in virtual storage as an alternative to repeatedly retrieving data from DASD.
* Certain IBM products or components such as LLA, TSO/E, CAS, and RACF use VLF data spaces as an alternate way to access data.
* A VLF class is a group of related objects made available to users of an application or component.

New MODIFY VLF command

* The new **MODIFY VLF,REPLACE,NN=** command was introduced in z/OS V2R1 to address the needs of installations that want to change the VLF configuration without having to restart VLF, which causes the loss of the current VLF cache and its performance benefit.
* Using the **MODIFY** command, the VLF configuration can be changed without needing to restart VLF, and the existing VLF cache is kept.

Usage and invocation

* The **MODIFY VLF,REPLACE,NN=** command is used to completely replace the current VLF configuration with a new one.
  - The Objects cached in VLF remain cached if that is still appropriate for a new configuration.

```
- MODIFY VLF,REPLACE,NN=01
COF026I MODIFY VLF PROCESSING IS COMPLETE.
```

NOTE: The replace commands supports a concatenation of up to 16 COFVLFxx parmlib members.

COFVLFxx parmlib members can be concatenated. When they are concatenated, a class definition can span parmlib members.

VLF PARMLIB concatenation

* z/OS V2R1 allows up to 16 COFVLFxx parmlib members to be concatenated to form one VLF configuration when VLF is started.
  - A concatenation of VLF configuration options allows more complex VLF configuration options.
  - This is a true concatenation because class definitions may span from one member to the next.

SMF record type 41 record, subtype 3, allows you to capture SMF data related to the usage of VLF.

Usage and invocation

* Up to 16 COFVLFxx parmlib members may be concatenated as described previously.
  - VLF treats the entire concatenation of parmlib members as one configuration.
  - Duplicate definitions will be rejected with the appropriate existing error messages.
  - When more than one COFVLFxx parmlib member is specified, the set of members should be enclosed in parenthesis and separated by commas.

VLF will not start unless SUB=MSTR is specified on the START command. SUB=MSTR means that VLF can continue to run across a JES restart. It is recommended that you arrange for the VLF start command to be issued automatically during the IPL process.

Start a concatenated VLF

```
-S VLF,SUB=MSTR,NN=(00,01)
. . .
COF011I  VLF  INITIALIZATION IS IN PROGRESS.
. . .
COF105I  COFVLF01. RECORD  28, CSVLLA IS A DUPLICATE CLASS DEFINTION.
COF105I  COFVLF01. RECORD  33, IKJEXEC IS A DUPLICATE CLASS DEFINTION.
COF105I  COFVLF01. RECORD  41, IGGCAS IS A DUPLICATE CLASS DEFINTION.
COF025I  VLF INITIALIZATION IS COMPLETED.
```

NOTE: If VLF is stopped and restarted during the life of an MVS IPL, any request by an application to retrieve or create a VLF object with a user token obtained prior to the restart causes VLF to pass a return code of X'10' back to the app and continue processing. The application should detect the return code and obtain a new token.

Possible COFVLFxx PARMLIB changes

Some considerations of the use of COFVLFxx concatenation are listed as following:

* Classes may be added or deleted.
  - When a class is deleted, any programs that are currently using it will receive existing failure return codes.
  - The cache for the class is purged.
* Major names (**EMAJ** or **EDSN**) may be added to or deleted from an existing class.
  - When a major name is deleted, any programs that are currently using it will receive existing failure return codes.
  - The objects in the cache associated with the deleted major are purged.
* The **MAXVIRT** parameter can be specified, raised, or lowered for an existing class.
  Note: A decrease in **MAXVIRT** may cause VLF trimming of the oldest objects to take place in order to reduce the cache to the new size.
* The **ALERTAGE** parameter can be specified, raised, or lowered for an existing class.
  Note: An ALERTAGE check parameter specified for the VLF Health Check will override this value even if it is changed via a MODIFY command.

Migration and coexistence considerations

* There isn't any APAR for pre-z/OS V2R1 releases to support a concatenation of COFVLFxx parmlib members

The *new* VLF Health Check

* The z/OS V2R1 introduce a new check for IBM Health Checker called **VLF_MAXVIRT** owned by **IBMVLF**.

Usage and invocation *owner (IBMVLF)*

* This new **VLF_MAXVIRT** check monitors VLF trimming activity to detect when objects are being trimmed that are younger than an age threshold, which can be an indication of thrashing because **MAXVIRT** is too low.
  - The **MAXVIRT** parameter specifies the data space size and the **ALERTAGE** specifies an age threshold.
* The **ALERTAGE** is an optional class parameter in COFVLFxx having a default value of 60 seconds.
  - The **ALERTAGE** can also be specified as a check parameter for the **VLF_MAXVIRT** check.
  - When it is specified as a check parameter, it *overrides* the value specified in COFVLFxx.
* The VLF_MAXVIRT check examines whether VLF is trimming recently added objects while making room for new objects.
  NOTE: Before this check, it was necessary to have the SMF 41(subtype 3) formatted in order to analyses the data spaces use.
  - The check parameter **ALERTAGE(class_name1,alert_age1,class_name2,alert_age2,...)** allows to define an alert age for objects being cached in VLF.
  - When an object younger than the alert value was trimmed, an exception is raised.
  - The class name allows wildcards, for example as in the default of **ALERTAGE(*,60)**, with alert age of 60 seconds, for all classes which do not have an **ALERTAGE** defined via COFVLFxx nor via this check parameter.
  - The **ALERTAGE** values defined in the check overrides the ones specified in the COFVLFxx.
    INTERVAL(1:00) - SEVERITY(LOW)

Below is an example of a successful VLF_MAXVIRT check (VLF_MAXVIRT)

```
CHECK (IBMVLF,VLF_MAXVIRT)
SYSPLEX:      PLEX05     SYSTEM: SC01
START TIME:   05/01/2014   12:46:54.456162
CHECK DATE: 20110802   CHECK SEVERITY:  LOW

COFVLH01I  For all classes, VLF is trimming objects within the goals set for this check.

END TIME: 05/01/2014 12:46:54.456316  STATUS: SUCCESSFUL
```

NOTE: If the **ALERTAGE** parameter for class CSVLLA is set, for example, to 60 seconds but VLF finds that it has trimmed an object for CSVLLA that was cached for only 45 seconds, the HealthCheck raises an alert to recommend that the **MAXVIRT** parameter be increased in order to provide more cache space for the CSVLLA class.

New syntax format of COFVLFxx

```
CLASS NAME(classname)
 {EDSN(dsn1) [VOL(vol)] EDSN(dsn2)...}
 {EMAJ(majname1) EMAJ(majname2)...}
 [MAXVIRT(nnn)]
 [ALERTAGE(alert_age)]
```

To have VLF stage load modules from LLA-managed libraries, you can use the default COFVLFxx member that is shipped in parmlib (COFVLF00), or, optionally, an installation-supplied COFVLFxx member.

ALERTAGE specifies the age, in seconds, for objects in the specified class. The IBM Health Checker for z/OS check IBMVLF, VLF_MAXVIRT uses the ALERTAGE to determine whether objects are being trimmed too rapidly to meet the installation's usage goals for VLF. You can specify an ALERTAGE in the range from 0 (which indicates that the check will find no exceptions for this class) to 99999999.

Default Value: ALERTAGE(60)Note that the higher the ALERTAGE value you specify, the more likely it is that the VLF_MAXVIRT check will issue an exception message.