

z/OS
3.2

*Integrated Accelerator for zEnterprise
Data Compression (zEDC)*



Note

Before using this information and the product it supports, read the information in [“Notices” on page 79.](#)

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 2019, 2025.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures.....	v
Tables.....	vii
About the Integrated Accelerator for zEnterprise Data Compression comprehensive content collection (c3).....	ix
How to provide feedback to IBM.....	xi
Summary of changes.....	xiii
Summary of Changes for z/OS 3.1.....	xiii
Chapter 1. What is Integrated Accelerator for zEnterprise Data Compression?.....	1
Chapter 2. Overview and planning of zEnterprise Data Compression (zEDC).....	3
Requirements for zEnterprise Data Compression.....	4
Product Enablement for zEnterprise Data Compression.....	4
Planning for zEnterprise Data Compression.....	5
Chapter 3. z/OS data.....	7
Request compression at the data set level.....	7
Request compression at the system level.....	8
Migration and backup and processing of dump data.....	9
SETSYS command: Establishing or changing the values of DFSMSHsm control parameters.....	9
DEFINE command: Defining control structures for use by DFSMSHsm.....	12
Assess compression of z/OS data.....	13
Record type 30 (X'1E') – Common address space work.....	14
RMF EADM reporting.....	18
Chapter 4. z/OS File System data.....	31
The compression process.....	31
Defining a new file system that is always compressed.....	32
zfsadm compress.....	32
Monitoring and displaying the compression status.....	33
Assess compression of zFS data.....	34
Chapter 5. z/OS applications.....	35
Displaying PCIE-related parameters (IQP).....	35
Assess compression of z/OS applications.....	36
Record type 113 (X'71') – Hardware capacity, reporting, and statistics.....	36
Chapter 6. z/OS databases.....	43
Chapter 7. Network transmission.....	45
Chapter 8. Programming.....	47
zEnterprise Data Compression (zEDC).....	47

Application interfaces for zEnterprise Data Compression.....	47
Programming: RMF.....	68
ERBSCMG3 - Extended Asynchronous Data Mover (EADM) data table.....	68
EADM - Tabular report data table ERBSCMT3.....	70
Chapter 9. Messages.....	73
RMF (ERB and GPM) messages.....	73
IQP messages.....	73
Appendix A. Accessibility.....	77
Notices.....	79
Terms and conditions for product documentation.....	80
IBM Online Privacy Statement.....	81
Policy for unsupported hardware.....	81
Minimum supported hardware.....	81
Trademarks.....	82
Index.....	83

Figures

1. Compression and Decompression Workflow.....3

2. EADM Activity Report..... 20

3. Extended Asynchronous Data Mover (EADM) Activity Report..... 22

Tables

1. Comparison table between unauthorized and z System authorized interfaces for zEDC.....	5
2. Fields in the EADM Activity Report.....	20
3. Fields in the EADM Activity Report.....	22
4. I/O Queuing Activity - Conditions Based on SMF Record Type 78-3.....	25
5. Standard zlib functions and whether they are supported using zEDC.....	48
6. Compression and decompression with zlib.....	52
7. Compression and decompression with z System authorized interfaces for zEDC.....	52
8. Environment for the FPZ4RZV service.....	53
9. Parameters for the FPZ4RZV service.....	53
10. Return and reason codes for the FPZ4RZV service.....	55
11. Environment for the FPZ4PRB service.....	56
12. Parameters for the FPZ4PRB service.....	56
13. Return and Reason Codes for the FPZ4PRB service.....	57
14. Environment for the FPZ4RMR service.....	57
15. Parameters for the FPZ4RMR service.....	58
16. Return and Reason Codes for the FPZ4RMR service.....	58
17. Environment for the FPZ4DMR service.....	60
18. Parameters for the FPZ4DMR service.....	60
19. Return and Reason Codes for the FPZ4DMR service.....	61
20. Environment for the FPZ4ABC service.....	61
21. Parameters for the FPZ4ABC service.....	62
22. Header elements in the FPZ4ABC-generated list.....	63
23. Entries elements in the FPZ4ABC-generated list.....	63

24. Return and Reason Codes for the FPZ4ABC service.....	64
25. Environment for the FPZ4URZ service.....	66
26. Parameters for the FPZ4URZ service.....	66
27. Return and Reason Codes for the FPZ4URZ service.....	67

About the Integrated Accelerator for zEnterprise Data Compression comprehensive content collection (c3)

Purpose of this information

This information is a collection of product documentation that you need to understand and use Integrated Accelerator for zEnterprise Data Compression (zEDC). Some of the information contained in this collection also exists elsewhere in the z/OS library.

Who should read this information

This information is intended for anyone who might implement or use compression, including system programmers, application developers, database administrators, and network administrators.

Related information

For an interactive starting point, and access to various technical resources about Integrated Accelerator for zEDC, see [Integrated Accelerator for zEDC \(www.ibm.com/support/z-content-solutions/compression/\)](http://www.ibm.com/support/z-content-solutions/compression/).

To find the complete z/OS library, go to [IBM Documentation \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. For more information, see [How to send feedback to IBM](#).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy) (www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy).

Summary of Changes for z/OS 3.1

The following content is new, changed, or no longer included in z/OS 3.1.

New

The following content is new.

April 2024 refresh

- The [Overview and planning of zEnterprise Data Compression \(zEDC\)](#) section is moved from [zEnterprise Data Compression \(zEDC\)](#) to its own chapter. It also includes a new subsection, [Product Enablement for zEnterprise Data Compression](#).

Changed

The following content is changed.

April 2024 refresh

- None.

Deleted

The following content is deleted.

April 2024 refresh

- None.

Chapter 1. What is Integrated Accelerator for zEnterprise Data Compression?

Integrated Accelerator for zEnterprise Data Compression (zEDC) with the IBM z15 replaces the zEDC Express adapter with on-chip compression, providing increased throughput and capacity. It reduces the cost of storing, processing, and transporting data.

The zEDC Express software support is a priced feature of z/OS that requires product enablement.

With it, you can enable compression for:

- Storage, including:
 - z/OS data (SMF logstreams, BSAM and QSAM data sets, DFSMSHsm and DFSMSdss processing)
 - z/OS File System (zFS) data
- z/OS applications, by using standard Java packages or zlib APIs
- z/OS databases (Db2 LOBs, Db2 archive logs, and Content Manager OnDemand)
- Network transmission with Sterling Connect:Direct.

For an interactive starting point, and access to various technical resources about Integrated Accelerator for zEDC, see [Integrated Accelerator for zEDC \(www.ibm.com/support/z-content-solutions/compression/\)](http://www.ibm.com/support/z-content-solutions/compression/).

Chapter 2. Overview and planning of zEnterprise Data Compression (zEDC)

In today's z/OS environment, many installations want to compress certain types of data to occupy less space while it is not in use, and then restore the data when necessary. Using zEnterprise® Data Compression (zEDC) to compress data might help to reduce CPU cost and elapsed time of data compression compared to traditional software-based compression services, such as CSRCESRV and CSRCMPSC. zEDC can also decrease the cost of applications that use host-based compression that are running on z/OS.

zEDC supports the DEFLATE compression data format, which compresses data by using the following algorithms, which are defined by RFC 1951:

- LZ77
 - Replaces repeated string with length, back pointer pairs.
 - Points back up to 32 K.
- Huffman coding
 - Variable length encoding of characters.
 - Minimize bit length of a stream of characters by assigning shorter codes to more frequent characters.
 - Data and length, back pointer pairs are Huffman encoded.

For more details, check the IETF standard RFC 1951 (tools.ietf.org/html/rfc1951).

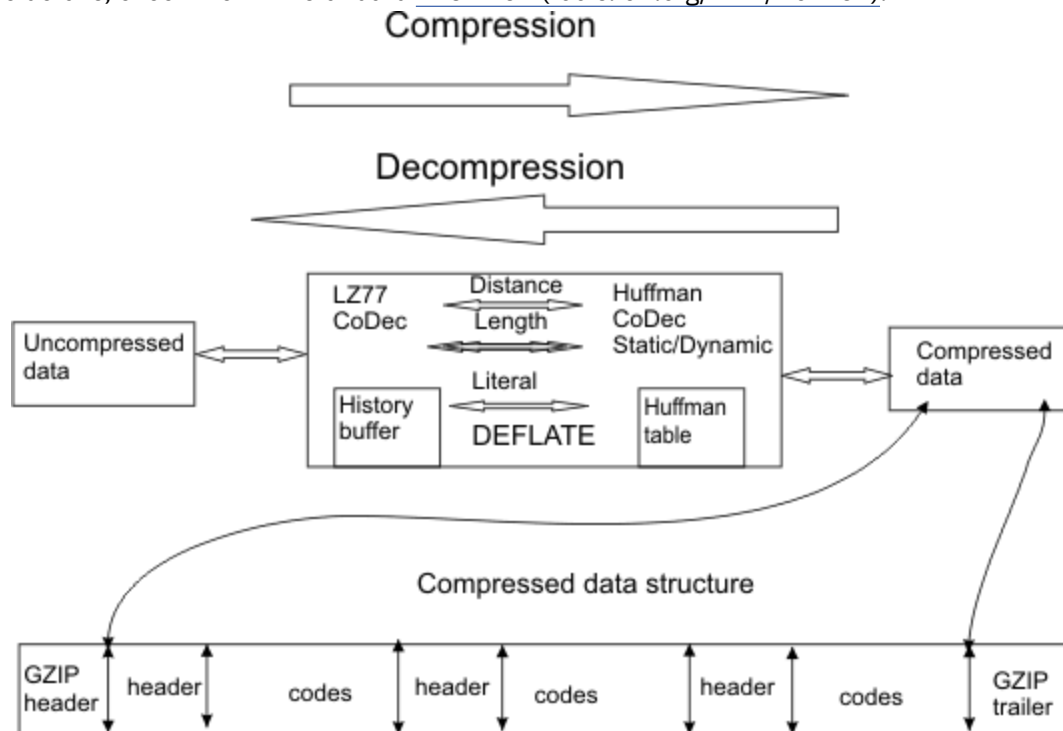


Figure 1. Compression and Decompression Workflow

For help with getting started, and access to various technical resources about Integrated Accelerator for zEDC, see [Integrated Accelerator for zEDC \(www.ibm.com/support/z-content-solutions/compression/\)](https://www.ibm.com/support/z-content-solutions/compression/).

Requirements for zEnterprise Data Compression

Hardware and software requirements.

zEDC requires the following:

- z/OS V2R1 (or later) operating system.
- One of the following:
 - IBM z15®, or later, with the Integrated Accelerator for zEDC
 - IBM zEnterprise EC12 CPC (with GA2 level microcode) or zBC12 CPC, or later, with the zEDC Express feature.
- zEDC software feature enabled in an IFAPRDxx parmlib member. For more information, see [“Product Enablement for zEnterprise Data Compression” on page 4](#).
- Adequate 64-bit real storage that is configured to this z/OS image.

Product Enablement for zEnterprise Data Compression

Enabling the priced software feature.

The IFAPRDxx PARMLIB member contains definitions for products that require product enablement, this includes zEDC. For more information about IFAPRDxx PARMLIB member, see [IFAPRDxx \(product enablement policy\)](#) in the *z/OS MVS Initialization and Tuning Reference*.

After receiving access to this priced software feature, you are required to change the IFAPRDxx PARMLIB member to include the following statements:

```
PRODUCT OWNER('IBM CORP')
NAME('Z/OS')
ID(5655-ZOS)
FEATURENAME(ZEDC)
VERSION(*)
RELEASE(*)
MOD(*)
STATE(ENABLED)
```

Note: When the member is updated, an IPL is required for the zEDC Express device driver to recognize the enablement.

The **STATE** parameter value should be set to disabled if the zEDC feature is no longer required.

If you try to dynamically enable the feature by using SET PROD=03 IBM MVS System Command, you see the following output:

```
IFA100I IN PARMLIB MEMBER=IFAPRD03 ON LINE 44
PRODUCTS WITH OWNER=IBM CORP NAME=Z/OS
FEATURE=ZEDC VERSION=*.** ID=5655-ZOS
HAVE BEEN ENABLED.
```

Otherwise, the status of the zEDC product feature remains Disabled. You can verify that the function is disabled with the DISPLAY IQP MVS system command, as shown in the following output:

```
D IQP
IQP066I 14.17.39 DISPLAY IQP
zEDC Information
DEFMINREQSIZE:          N/A
INFINREQSIZE:           N/A
Feature Enablement:     Disabled
```

After an IPL, you can verify that the zEDC product feature is Enabled by issuing the DISPLAY IQP MVS system command.

```
D IQP
IQP066I 14.58.45 DISPLAY IQP
zEDC Information
DEFMINREQSIZE:          1K (STATIC)
```

INFMINREQSIZE: 1K (STATIC)
Feature Enablement: Enabled

You can also use the D PROD,REG,FEATURENAME(ZEDC) MVS system command to verify the status of the priced feature. The following output from this command is shown.

```
IFA111I 16.25.49 PROD DISPLAY
S OWNER      NAME      FEATURE  VERSION  ID
E IBM CORP   z/OS      ZEDC     03.01.00 5655-Z0S
```

Planning for zEnterprise Data Compression

Planning for zEnterprise Data Compression use.

zEDC is established by starting either an unauthorized or authorized interface:

- Unauthorized interface for zEDC:
 - zlib for zEDC:
 - zlib is an OpenSource data compression library that supports the DEFLATE compressed data format.
 - The zlib compression library provides in-memory compression and decompression functions, including integrity checks of the decompressed data. For more information, see [zlib Compression Library \(zlib.net\)](#).
- zSystem authorized interfaces for zEDC:
 - Requires supervisor state and supports task and SRB mode.
 - Allows application buffers to be directly read by and written to by compression accelerator hardware, allowing the application to avoid a data move, but also adding complexity to managing I/O buffers.
 - Operates on independent requests:
 - A deflate request produces a full DEFLATE block.
 - An inflate request consumes a full DEFLATE block.
 - Provides software inflate capability to maintain data access when z System compression accelerator hardware is not available.
- Additional method with the option to use zEDC:
 - SMF compression. Use the COMPRESS and PERMFIX keywords in the SMFPRMxx parmlib member to compress data before writing to a log stream. For more information, see [z/OS MVS System Management Facilities \(SMF\)](#) and [z/OS MVS Initialization and Tuning Reference](#).

Table 1. Comparison table between unauthorized and z System authorized interfaces for zEDC		
Options	Unauthorized interfaces for zEDC	zSystem authorized interfaces for zEDC
Language	C	Any language that can call OS callable services
Data streaming	zlib-style data streams supported. Data can be broken up across requests as needed, but must be within the minimum input buffer limit.	Each request is independent and handled as a single DEFLATE block. Inflate requests must receive single complete DEFLATE block.
Buffer management	Data move to device driver-managed buffer (IBM z14® and earlier) or data that is compressed directly by using the buffer in the application (IBM z15 and higher).	Application buffer directly used by z System hardware.

<i>Table 1. Comparison table between unauthorized and z System authorized interfaces for zEDC (continued)</i>		
Options	Unauthorized interfaces for zEDC	zSystem authorized interfaces for zEDC
Co-existence support	Both inflate and deflate are completed in software when hardware is not available.	Inflate completed in software when hardware is not available.
Authorization	Controlled by SAF-protected FACILITY class resource FPZ.ACCELERATOR.COMPRESSION (IBM z14 and earlier).	Supervisor state.

Chapter 3. z/OS data

Use z/OS data compression for SMF logstreams, BSAM and QSAM data sets, and DFSMSHsm and DFSMSdss processing.

If you have been using zEDC with the zEDC Express adapter, review your definitions for restrictions that you might remove, to apply compression more broadly. For example:

- Compression of sequential data sets that is selectively enabled by application
- For migration to tape, balancing zEDC against data set compression. Consider removing any restrictions on what you compress.

If you have not been using zEDC, plan for and implement compression:

- At the data set level, using the COMPACTION option of a data class.
- At the system level, using the COMPRESS parameter in the IGDSMSxx member of parmlib, which defines options for the storage management subsystem (SMS)
- During DFSMSHsm and DFSMSdss processing, using the ZCOMPRESS parameter of the SETSYS command
- Of dump data, using the ZCOMPRESS parameter of the DEFINE DUMPCLASS command.

Space requirements: To allocate a zEDC compressed format data set, the allocation amount must also meet this minimum space requirement:

- 5 MB minimum primary space amount if a secondary amount is specified
- 8 MB if a secondary amount is not specified.

If the minimum space requirement is not met, the allocation request might result in a non-compressed extended format data set.

Assessing compression: After your definitions are complete and your applications have run, you can assess compression with:

- SMF record type 30
- RMF EADM reporting (RMF 74.10).

For more information, see [“Assess compression of z/OS data” on page 13.](#)

Request compression at the data set level

You request compression at the data set level by defining a data class with the compaction options ZR (zEDC Required) or ZP (zEDC Preferred).

The compaction option on the DFSMS Data Class Define panel is as follows:

Compaction

Specifies whether the data set is to be compressed format. You can compress data that is being written to an extended format data set or to a PDSE. This field is ignored for DASD data sets if the data set name type is not EXT or LIBRARY. You can compress data when it is being written to tape. A compressed data set cannot reside on the same tape cartridge as a data set that is not compressed.

The possible values are:

Y

Extended format or PDSE data sets are compressed and tape volumes are compacted. The type of DASD compression depends on the COMPRESS option in parmlib member IGDSMSxx. Tape volumes are compacted unless compaction is overridden by the user through JCL/dynamic allocation.

N

Data sets are not compressed. Tape volumes are not compacted, unless compaction is requested by the user through JCL/dynamic allocation.

T

Extended format data sets are compressed using tailored dictionaries. This overrides the COMPRESS option in parmlib member IGDSMSxx.

G

Extended format data sets are compressed using generic dictionaries. This overrides the COMPRESS option in parmlib member IGDSMSxx.

ZR

Indicates "zEDC Required", meaning that the system should fail the allocation request if the zEDC function is not supported by the system, or if the minimum allocation amount requirement is not met.

ZP

Indicates "zEDC Preferred", meaning that the system should not fail the allocation request, but rather create either a tailored compressed data set if the zEDC function is not supported by the system, or a non-compressed extended format data set if the minimum allocation amount requirement is not met.

blank

Data sets are not compressed. Tape volumes might be compacted depending on what was specified by the user on JCL/dynamic allocation, the installation with the COMPACT option in parmlib member DEVSUPxx, or the allocated hardware model. This is the default.

T, G, ZR, and ZP override the compression option set in the IGDSMSxx PARMLIB member, and let you select the type of compression on a data set level. Current users of generic compression can move to using tailored or zEDC compression one data set at a time, as new data sets are created.

Request compression at the system level

You request compression at the system level by defining the storage management subsystem (SMS) with the COMPRESS options ZEDC_R (zEDC Required) or ZEDC_P (zEDC Preferred) in the IGDSMSxx member of parmlib. The data class COMPACTION value must be set to Y.

The COMPRESS option is as follows:

COMPRESS({TAILORED|GENERIC|ZEDC_R|ZEDC_P})

Specifies the type of compression to be used for the data set.

GENERIC

Specifies that the data set be compressed using generic Dictionary Building Block (DBB) compression. The dictionary is derived from a defined set of compression algorithms in data set SYS1.DBBLIB.

TAILORED

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

To convert an existing DBB-based compressed data set to use tailored compressions, you must set the COMPRESS parameter to TAILORED and copy the generic DBB-based data set to a new data set that meets compression requirements.

ZEDC_P

Specifies that the data set be compressed using zEDC compression, as with the ZEDC_R option. But here, the system does not fail the allocation request, but rather create either a tailored compressed data set if the zEDC function is not supported by the system, or create a non-compressed extended format data set if the minimum allocation amount requirement is not met.

ZEDC_R

Specifies that the data set be compressed using zEnterprise data compression (zEDC). No separate dictionary is created, as zEDC compression hides the dictionary in the data stream. A new dictionary starts in each compression unit. The system can decompress the segment as is.

With this option (as opposed to ZEDC_P), the system fails the allocation request if the zEDC function is not supported by the system, or if the minimum allocation amount requirement is not met.

Note:

1. Use tailored compression only when all systems in the SMS complex have been converted to DFSMS/MVS 1.4 or later, and when there is no need to revert to a prior release level for local recovery or remote recovery with Aggregate Backup and Recovery Support (ABARS). The date of general availability for DFSMS/MVS 1.4 was in 1997.
2. To convert an existing DBB-based (generic) compressed data set to use tailored compression, or to convert from either generic or tailored to zEDC compression, first set the COMPRESS parameter to TAILORED, ZEDC_R, or ZEDC_P in the IGDSMSxx parmlib member. Use IEBGENER, ICEGENER, REPRO, or any QSAM or BSAM application to copy the data set to a new data set that meets compression requirements.
3. The zEDC feature must be installed on all systems before using zEDC compression. For more information on zEDC compression, see [z/OS MVS Programming: Callable Services for High-Level Languages](#).

Default: GENERIC

For a description of the data class COMPACTION option, see [“Request compression at the data set level” on page 7](#).

Migration and backup and processing of dump data

You request compression during DFSMSHsm migration and backup and processing of dump data, with the ZCOMPRESS parameter of the SETSYS command and the ZCOMPRESS parameter of the DEFINE DUMPCLASS command.

The ZCOMPRESS parameter on the SETSYS and DEFINE DUMPCLASS commands is described in the topics that follow.

SETSYS command: Establishing or changing the values of DFSMSHsm control parameters

When you start DFSMSHsm, a subset of DFSMSHsm control parameters is established by default. You can override DFSMSHsm defaults by specifying one or more SETSYS commands in the ARCCMDxx PARMLIB member used when you start DFSMSHsm. You can issue the SETSYS command with specific parameter values after DFSMSHsm is started to change the current defaults. The changed values remain in effect until you restart DFSMSHsm. For an example of how to set up the ARCCMDxx PARMLIB member, see the DFSMSHsm installation verification procedure in *z/OS Program Directory* in the [z/OS Internet library](http://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

If you do not specify the SETSYS command, DFSMSHsm does not do any automatic space management, backup, or dump. Therefore, if you want to take advantage of the automatic functions of DFSMSHsm, use the SETSYS command.

Do not confuse DFSMSHsm defaults with the SETSYS command defaults. Except for certain values, there are no SETSYS command defaults. A SETSYS command does not have required parameters, so unless you indicate a specific parameter value for the SETSYS command, the DFSMSHsm control parameter is the default. An exception to this is the CDSVERSIONBACKUP parameter.

The following link provides a deep dive into the many knobs and dials available to configure DFSMSHsm uniquely to your environment.

DFSMSHsm Education Series-hsm SETSYS (mediacenter.ibm.com/media/DFSMSHsm%20Education%20Series-hsm%20SETSYS/1_qh3d8942)

Note: Although the SETSYS command has no required parameters, you must specify at least one of the optional parameters to change any system parameters or default values.

ZCOMPRESS (NONE | ALL): Specifying whether to use zEDC Services for data sets

Explanation: **ALL** | **NONE** are mutually exclusive, optional subparameters of the ZCOMPRESS parameter, specifying whether or not to use zEDC Services during migration or backup.

Subparameter	Explanation
NONE	DFSMSHsm does <i>not</i> use zEDC for a data set during migration or backup. DFSMSHsm will use the SETSYS COMPACT settings.
ALL	DFSMSHsm will use zEDC during migration and backup, if this function is supported by the system. If zEDC Services are <i>not</i> available, DFSMSHsm will use the SETSYS COMPACT settings to determine what type of software compaction is used for migration and backup.

SMS relationship: Parameter differs in meaning when applied to SMS-managed or non-SMS-managed DASD volumes or data sets.

SETSYS default: None.

DFSMSHsm default: If you do not specify any subparameter of the ZCOMPRESS parameter on any SETSYS command, the DFSMSHsm default is NONE and the SETSYS COMPACT settings will be used.

Note:

1. DFSMSHsm does *not* use zEDC for partitioned data sets.
2. When you specify the ALL or NONE subparameter with the ZCOMPRESS parameter, DFSMSHsm ignores all other subparameters.

ZCOMPRESS (DASDBACKUP (NO | YES)) : Specifying whether to use zEDC Services for data sets during backup to DASD

Explanation: **DASDBACKUP (NO | YES)** are mutually exclusive, optional subparameters of the ZCOMPRESS parameter, specifying whether or not to use zEDC Services when DFSMSHsm backs up to DASD.

Subparameter	Explanation
DASDBACKUP (NO)	DFSMSHsm does <i>not</i> use zEDC for a data set backup to DASD. DFSMSHsm will use the SETSYS COMPACT settings.
DASDBACKUP (YES)	DFSMSHsm will use zEDC Services during backup to DASD, if this function is supported by the system. If zEDC is <i>not</i> available, DFSMSHsm will use the SETSYS COMPACT settings to determine what type of software compaction is used for backup.

SMS relationship: Parameter differs in meaning when applied to SMS-managed or non-SMS-managed DASD volumes or data sets.

SETSYS default: None.

DFSMSHsm default: If you do not specify either subparameter on any SETSYS command, the DFSMSHsm default is DASDBACKUP(NO).

Note:

1. If you do not want a specific data set to be compressed with zEDC during volume backup to DASD, use the data set backup exit (ARCBDEXT) to prevent compression.
2. As compressed-format data sets are already compressed, they are not compressed with zEDC Services during backup to DASD.
3. Since encrypted data sets do not compress, zEDC compression services will not be used during backup to DASD.

ZCOMPRESS (DASDMIGRATE (NO | YES)) : Specifying whether to use zEDC Services for data sets during migration to DASD

Explanation: **DASDMIGRATE (NO | YES)** are mutually exclusive, optional subparameters of the ZCOMPRESS parameter, specifying whether or not to use zEDC Services when DFSMSHsm migrates to DASD.

Subparameter	Explanation
DASDMIGRATE (NO)	DFSMSHsm does <i>not</i> use zEDC for a data set migration to DASD. DFSMSHsm will use the SETSYS COMPACT settings.
DASDMIGRATE (YES)	DFSMSHsm will use zEDC Services during migration to DASD, if this function is supported by the system. If zEDC is <i>not</i> available, DFSMSHsm will use the SETSYS COMPACT settings to determine what type of software compaction is used for migration.

SMS relationship: Parameter differs in meaning when applied to SMS-managed or non-SMS-managed DASD volumes or data sets.

SETSYS default: None.

DFSMSHsm default: If you do not specify either subparameter on any SETSYS command, the DFSMSHsm default is DASDMIGRATE(NO).

Note:

1. If you do not want a specific data set to be compressed with zEDC during volume migration to DASD, use the data set migration exit (ARCMDEXT) to prevent compression.
2. As compressed-format data sets are already compressed, they are not compressed with zEDC Services during migration to DASD.
3. Since encrypted data sets do not compress, zEDC compression services will not be used during migration to DASD.

ZCOMPRESS (TAPEBACKUP (NO | YES)) : Specifying whether to use zEDC Services for data sets during backup to tape

Explanation: **TAPEBACKUP (NO | YES)** are mutually exclusive, optional subparameters of the ZCOMPRESS parameter, specifying whether or not to use zEDC Services when DFSMSHsm backs up to tape.

Subparameter	Explanation
TAPEBACKUP (NO)	DFSMSHsm does <i>not</i> use zEDC for a data set backup to tape. DFSMSHsm will use the SETSYS COMPACT settings.
TAPEBACKUP (YES)	DFSMSHsm will use zEDC Services during backup to tape, if this function is supported by the system. If zEDC is <i>not</i> available, DFSMSHsm will use the SETSYS COMPACT settings to determine what type of software compaction is used for backup.

SMS relationship: Parameter differs in meaning when applied to SMS-managed or non-SMS-managed DASD volumes or data sets.

SETSYS default: None.

DFSMSHsm default: If you do not specify either subparameter on any SETSYS command, the DFSMSHsm default is TAPEBACKUP(NO).

Note:

1. If you do not want a specific data set to be compressed with zEDC during volume backup to tape, use the data set backup exit (ARCBDEXT) to prevent compression.
2. As compressed-format data sets are already compressed, they are not compressed with zEDC Services during backup to tape.
3. Since encrypted data sets do not compress, zEDC compression services will not be used during migration to DASD.

ZCOMPRESS (TAPEMIGRATE (NO | YES)) : Specifying whether to use zEDC Services for data sets during migration to tape

Explanation: **TAPEMIGRATE (NO | YES)** are mutually exclusive, optional subparameters of the ZCOMPRESS parameter, specifying whether or not to use zEDC Services when DFSMSHsm migrates to tape.

Subparameter	Explanation
TAPEMIGRATE (NO)	DFSMSHsm does <i>not</i> use zEDC for a data set migration to tape. DFSMSHsm will use the SETSYS COMPACT settings.
TAPEMIGRATE (YES)	DFSMSHsm will use zEDC Services during migration to tape, if this function is supported by the system. If zEDC is <i>not</i> available, DFSMSHsm will use the SETSYS COMPACT settings to determine what type of software compaction is used for migration.

SMS relationship: Parameter differs in meaning when applied to SMS-managed or non-SMS-managed DASD volumes or data sets.

SETSYS default: None.

DFSMSHsm default: If you do not specify either subparameter on any SETSYS command, the DFSMSHsm default is TAPEMIGRATE(NO).

Note:

1. If you do not want a specific data set to be compressed with zEDC during volume migration to tape, use the data set migration exit (ARCMDEXT) to prevent compression.
2. As compressed-format data sets are already compressed, they are not compressed with zEDC Services during migration to tape.
3. Since encrypted data sets do not compress, zEDC compression services will not be used during migration to tape.

DEFINE command: Defining control structures for use by DFSMSHsm

The DEFINE command allows you to specify the following control structures that are used by DFSMSHsm:

- Backup cycle, start date for the backup cycle, and the number of daily backup volumes to be used for each day in the backup cycle
- Dump class attributes
- Dump cycle and start date for the dump cycle
- Automatic primary space management cycle
- Automatic secondary space management cycle
- Key ranges for DASD migration level 2 volumes

- Pool of volumes to which non-SMS-managed data sets with the same set of initial characters are recalled
- Pool of volumes, that together can be used as a target for recalling a data set migrated from one of the volumes
- Pool of volumes to which backed-up aggregated data sets will be restored during aggregate recovery
- When to release a data set backup tape that is mounted to process data set backup commands and the status of the released tape.

When you use JES3, you must use the ADDVOL command to add primary volumes to DFSMSHsm before you can specify the volumes with the POOL or the VOLUMEPOOL parameter of the DEFINE command.

If an SMS-managed volume is specified as part of a data set pool, a message is issued indicating that the volume has not been added to the data set pool. Processing continues for the DEFINE command. If, however, only SMS-managed volumes are specified for the definition of the data set pool, the data set pool definition is rejected, and the DEFINE command fails.

With the exception of the ARPOOL, POOL and VOLUMEPOOL parameters, you do not have to redefine any optional parameters each time you start DFSMSHsm.

DUMPClass(ZCOMPRESS): Specifying whether to use zEDC Services on the dump data

Explanation: **ZCOMPRESS** is an optional subparameter specifying whether DFSMSdss is to use zEDC Services on the dump data.

Subparameter	Explanation
ZCOMPRESS(YES)	<p>DUMPClass(ZCOMPRESS(YES)) When YES is specified, compression with zEDC will be performed, if this function is supported by the system. If zEDC Services are <i>not</i> available, the other compression type will be used.</p> <p>For a CLOUD dump class, CDA compression with zEDC Services will be used when a DIRECT cloud provider is specified in the CLOUD parameter.</p>
ZCOMPRESS(NO)	<p>DUMPClass(ZCOMPRESS(NO)) When NO is specified, compression with zEDC will <i>not</i> be performed. ZCOMPRESS(NO) can be specified to override an existing setting of ZCOMPRESS(YES).</p> <p>For a CLOUD dump class, CDA compression will not be used when a DIRECT cloud provider is specified in the CLOUD parameter.</p>

Defaults: When ZCOMPRESS is not specified and was not previously set in the existing dump class definition, NO is used by default.

Assess compression of z/OS data

After your definitions are complete and your applications have run, you can assess compression of z/OS data with SMF records and RMF reports.

For performance metrics use:

- SMF record type 30
- RMF EADM reporting (RMF 74.10).

The SMF records and RMF reports are described in the topics that follow. For more information about SMF, see [z/OS MVS System Management Facilities \(SMF\)](#).

For more information about RMF reports, see [z/OS Resource Measurement Facility User's Guide](#).

Record type 30 (X'1E') – Common address space work

SMFPRMxx parameters are described in [SMFPRMxx \(system management facilities \(SMF\) parameters\)](#) in *z/OS MVS Initialization and Tuning Reference*.

Information about system address spaces and full function start are described in [Creating address spaces for system components](#) in *z/OS MVS Initialization and Tuning Guide*.

The type 30 SMF record provides accounting information. It consolidates data that is found in record types 4, 5, 20, 34, 35, and 40 (which simplifies accounting by installation-written post processing routines), and it provides additional information. Use record type 30, because the record types that it replaces are generally not being updated with new measurement data. SMF writes record type 30 when:

- A work unit (such as a TSO/E session, APPC/MVS transaction program, OMVS forked or spawned address space, started task, or batch job) starts. This subtype 1 record identifies the work unit but contains no resource data.
- An SMF interval ends, if you requested interval accounting.

If this is the first interval since the work unit started, then this subtype 2 record contains the total resources used from the start of the work unit until the end of the current interval. With interval synchronization, this span of time is normally shorter than the length of the SMF global recording interval. For global interval recording without interval synchronization, this span of time is the same as the length of the SMF global recording interval.

For other intervals, this subtype 2 record contains the total resources used from the end of the previous interval until the end of the current interval.

For system address spaces that do not go through full function start, SMF generates a subtype 6 record that contains the total resources used since the start of the address space. Note that the data in the subtype 6 record is cumulative, unlike the subtype 2 record.

- A work unit (such as a TSO/E session, APPC/MVS transaction program, OMVS forked or spawned address space, started task, or batch job) completes.

If you requested interval accounting, SMF generates a subtype 3 record that contains the total resources used from the end of the previous recording interval until the end of the work unit. This span of time is normally shorter than the length of the specified recording interval.

For a job step, SMF generates a subtype 4 record that contains the total resources used from the time when the job step started until the time when the job step completed. If you requested interval recording, then this subtype 4 record generally contains the accumulated totals of the data in the interval subtype 2 and subtype 3 records that were generated for the step.

For a job, SMF generates a subtype 5 record that contains the total resources used from the time when the job started until the time when the job completed. This subtype 5 record generally contains the accumulated totals of the data in the step total subtype 4 records that were generated for the job.

The type 30 record contains operation information such as the job and step start and end times, step CPU time, step termination status, number of records in DD DATA and DD * data sets processed by the step and job, device allocation start time, problem program start time, and storage protect key. The record contains the number of page-ins, page-outs, swap-ins, and swap-outs for both virtual input output (VIO) and non-VIO data sets. The record contains information on the number of hiperspace page moves and the movement of pages between expanded storage and central storage. This data can be used in resource planning. Information is added to account for time spent in hiperspace processing on a step or interval basis. The record contains an entry for each data set defined by a DD statement or dynamic allocation. Each entry lists the device class, unit type, device number, the execute channel program (EXCP) count, and device connect time for the data set. The usage data section contains information that can be used to attribute usage of a product to the address space.

The subtypes are:

Subtype	Meaning
---------	---------

- 1** Job start or start of other work unit
- 2** Activity since previous interval ended
- 3** Activity for the last interval before step termination
- 4** Step total
- 5** Job termination or termination of other work unit
- 6** System address space, which did not go through full function start. See [Creating address spaces for system components in z/OS MVS Initialization and Tuning Guide](#) for information about system address spaces. When you select subtype 6 for record type 30, the following fields may contain zeros or blank:
 - SMF30PGM
 - SMF30STM
 - SMF30PSN
 - SMF30CL8
 - SMF30UIF
 - SMF30USR
 - SMF30JNM

Information in specific fields may differ for different subtypes. For example, the record identifies the job (and job step) by the:

- Job log identification (job name, time and date that the reader recognized the job card for this job).
- Step name
- Number of the step within the job
- User identification
- Program name
- Performance group number or service class name
- JES job number.

If accounting numbers (which can be alphanumeric) are specified in the JOB or EXEC statements, they are included. For subtype 1 and subtype 5, the accounting numbers are taken from the JOB statement. For all other subtypes, the accounting numbers are taken from the EXEC statement.

Because some of the information necessary to complete a field is not always available when a type 30 record is written, some fields might be empty. For example, the SMF30AST, SMF30PPS, SMF30SIT, and SMF30STD fields are not filled in for a subtype 1 record.

Because system address spaces do not use full function start, the subtype 6 record is incomplete; that is, only certain fields in each section are valid. All unused fields are set to zero or blank.

The subtype 6 records are written only at the expiration of an interval; the values are cumulative and indicate data collected since the initialization of the address space. If a system address space later goes through full function start, data is not reported for the period between the expiration of the previous interval and the time that the address space goes through full function start. The subtype 6 record contains data for the APPC/MVS Cumulative Resource section, but data in the APPC/MVS Resource section is not reported in subtype 6.

The length of record type 30 is variable. The maximum length of the type 30 record is 32,756 bytes. If the volume of data in the type 30 record is such that the length would exceed the maximum length, one or more additional type 30 records are produced. The additional records contain only the header/self-defining, subsystem, identification, and one or more sections that can repeat. An example of a section that can repeat is the execute channel program (EXCP) section.

Rules for SMF type 30 continuation record processing: For all subtypes (except subtype 1), it is possible to have additional continuation records.

When the value in SMF30SOF is greater than 192 (X'C0'), the following record continuation rules apply:

- A record is a member of a group of continuation records if *any* of the following flags is on. A record is not a member of a group of continuation records if *all* of these flags are off.

SMF30_RecCont_FirstRec
SMF30_RecCont_AdditionalRec
SMF30_RecCont_LastRec

- When a record is a member of a group of continuation records but not the last record in the group, the value in SMF30_Cont_Recs_To_Follow is the number of continuation records that follow the current record in the group. For the last record in the group of continuation records, SMF30_RecCont_LastRec is on and SMF30_Cont_Recs_To_Follow contains a value of zero.
- The contents of the SMF type 30 Identification section (DSECT SMF30ID) is the same for all records in a group of continuation records.

When the value in SMF30SOF is less than or equal to 192 (X'C0'), the following record continuation rules apply:

- A record is the *first record* if at least one of the following fields is non-zero:

SMF30AON
SMF30ARN
SMF30CON
SMF30DRN
SMF30OON
SMF30PON
SMF30RON
SMF30TON
SMF30UON

- A record is an *additional record* if the following fields are all zero:

SMF30AON
SMF30ARN
SMF30CON
SMF30DRN
SMF30OON
SMF30PON
SMF30RON
SMF30TON
SMF30UON

- In either a *first* or *additional* record:

- There are more records to follow if at least one of the following fields is non-zero:

SMF30EOS
SMF30MOS
SMF30OPM
SMF30RMS
SMF30UDS

- This is the *last record* if the following fields are all zero:

SMF30EOS
SMF30MOS
SMF30OPM
SMF30RMS

SMF30UDS

The IEFACTRT exit is called at step and job termination for each type 30 (subtype 4 and subtype 5) record written to the SMF data set. A separate call to IEFACTRT is made for each additional record.

Notes:

1. Data sets are recorded in the order of the DD statements; they are not identified by name. However, the data definition name (ddname) is included in the record. (An installation-written IEFUJV exit routine can record this order as each statement is validated). For concatenated DD statements, the ddname is the same on each entry, respectively.
2. CPU time is not expected to be constant between different runs of the same job step.
3. If the SMFPRMxx parameter DDCONS(YES) is specified, then duplicate execute channel program (EXCP) entries are consolidated. If DDCONS(NO) is specified, then duplicate EXCP entries are not consolidated. SMFPRMxx parameters are described in *z/OS MVS Initialization and Tuning Reference*.
4. If a section is not included in the record, the “number of” entry is zero. For example, subtype 1 does not have a completion segment, and SMF30TON is set to zero to indicate this.
5. If the IEFUSI exit changes the size of the private area, a flag is set in SMF30SFL in the paging and storage section.
6. Specifying the DETAIL parameter of the SMFPRMxx parmlib member for STC includes all EXCP sections in subtypes 4 and 5 for the step or job.
7. Specifying the NODETAIL parameter excludes EXCP sections from subtypes 4 and 5 for STC class subsystem jobs, but not for batch or TSO/E subsystems. NODETAIL is enforced for the master address space.

For OMVS, the OMVS address space is considered a started task (STC), so NODETAIL is honored for processes running in the OMVS address space. However, processes that run under a BPXAS initiator are considered batch work, so NODETAIL has no effect.

8. Considerations for jobs that are evicted via the \$EJxx, STEP (or equivalent) command:
 - SMF30CNR (bit 14 of SMF30STI) is set in the subtype 4 record of the last step that executed prior to the eviction.
 - A subtype 5 record is not generated for a job that is evicted until the job resumes and completes execution.
 - Except for the following fields, subtype 5 fields that contain aggregated data will not include data from all steps in a job that resumes and completes execution following an eviction. In this case, subtype 5 fields will contain data aggregated from the steps that ran after the job resumes execution up until the job completes execution. In the subtype 5 record of an evicted job, these fields will contain data aggregated from all of the steps in the job:

SMF30CPT
SMF30CPS
SMF30_TIME_ON_IFA
SMF30_TIME_IFA_ON_CP
SMF30_TIME_ON_SUP
SMF30_TIME_SUP_ON_CP

9. While a job is executing, SMF writes the type 30, subtype 1, 4, and 5 records associated with each job step to the job's EVENTLOG data set. You can use the Job Step panel of the Spool Display and Search Facility (SDSF) to view these records. To control access to this information, the EVENTLOG is protected with two SAF resources in the JESSPOOL class: *nodeid.userid.jobname.jobid.EVENTLOG.SMFSTEP* and *nodeid.userid.jobname.jobid.EVENTLOG.STEPDATA*. For more information, see [Jobs, job groups, output groups, and SYSIN/SYSOUT data sets](#) in *z/OS SDSF Operation and Customization*.

Using the SMF30CPT field in the Accounting section

Note that a workload may generate different values for SMF30CPT, if some eligible work for the IBM zEnterprise Application Assist Processor (zAAP) or IBM z Integrated Information Processor (zIIP)

running on a standard processor. If a repeatable value is more desirable than the possible performance benefits of letting zAAP or zIIP eligible work run on both specialty CPs and standard processors, specify IFAHONORPRIORITY=NO or IIPHONORPRIORITY=NO in the IEAOPTxx parmlib member.

Interval records may show this number to be hundredth (1/100) of a second less than other related SMF30 fields (such as SMF30_TIME_IFA_ON_CP). This difference is due to rounding differentials while calculating delta values, and will not occur for step end or job end.

If SMF30CPT is zero, and you would like to understand how much CPU time was used by the address space or you require more precise values in general, you can calculate the CPU and SRB time in microseconds ($1 / 10^{-6}$) using the following formulas:

- **CPU time:**

$$(SMF30CSU \times 10) / SMF30CPC \times SMF30SUS / 16 = \text{CPU time in microseconds}$$

- **SRB time:**

$$(SMF30SRB \times 10) / SMF30SRC \times SMF30SUS / 16 = \text{SRB time in microseconds}$$

The CPU and SRB times computed from service units include zAAP time and zIIP time in addition to CP time. The time derived from CPU service is comparable to the sum of fields SMF30CPT, SMF30_TIME_ON_IFA and SMF30_TIME_ON_zIIP for work with no enclave activity. If the zAAP or zIIP processors are faster than the CP, zAAP or zIIP time is normalized to the time expected on the slower CP before service units are computed. Therefore, the derived time contains normalized zAAP or zIIP time. Field SMF30ZNF contains the normalization factor used. The normalization factor is 256 when the standard CPs are of the same speed as the zAAP or zIIP. Enclave time is summed with address space time in SMF30CPT. The preceding CPU time formula includes dependent enclave time but does not include independent enclave time. To compute independent enclave time, substitute SMF30ESU for SMF30CSU in the formula.

If you use derived processor times, you determine:

$$\begin{aligned} \text{derived CP time} &= CPT / (CPT + zAAPNT + zIIPNT) \times \text{derived CPU time} \\ \text{derived Normalized zIIP time} &= zIIPNT / (CPT + zAAPNT + zIIPNT) \times \text{derived CPU time} \\ \text{derived Normalized zAAP time} &= zAAPNT / (CPT + zAAPNT + zIIPNT) \times \text{derived CPU time} \end{aligned}$$

where:

$$\begin{aligned} CPT &= SMF30CPT \\ zAAPNT &= SMF30_TIME_ON_IFA \times (SMF30ZNF / 256) \\ zIIPNT &= SMF30_TIME_ON_zIIP \times (SMF30ZNF / 256) \end{aligned}$$

assuming the denominator is not zero.

RMF EADM reporting

With APAR OA56684, RMF is enhanced to report performance metrics for asynchronous compression and decompression activity. Compression and decompression requests are directed to the Extended Asynchronous Data Mover (EADM).

RMF Monitor I collects utilization information of IOP resources used by EADM-compression. The returned utilization information is stored into SMF 78 subtype 3 records. The RMF Postprocessor uses this to report the percentage of IOP Busy for EADM compression activity in the I/O Queuing Activity report.

In addition, with this support, the RMF Postprocessor and Monitor III SCM Activation reports are renamed to EADM Activity reports. Both reports continue to provide RMF users with SCM activity data.

The number of compress and decompress operations as well as the number of input and output bytes for asynchronous compression and decompression are stored in SMF 74 subtype 10 and into the Monitor III Set-of-Samples. These numbers are used by the RMF Postprocessor and Monitor III to calculate and display compression and decompression request rates, throughput and ratios in the new Compression

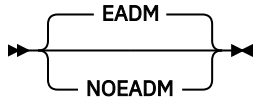
Activity report section of the Postprocessor EADM Activity report and in the Monitor III EADM Activity report header section.

To control the collection of EADM data, use the EADM Monitor III data gatherer option.

To control the reports to be generated by the Postprocessor for a single system, use the REPORTS control statement.

An IOPECB condition has been added to I/O Queuing Activity conditions based on SMF Record Type 78-3.

EADM Monitor III data gatherer option



Controls the collection of activity data for extended asynchronous data mover (EADM). The default is EADM.

Note: This gatherer option was formerly called SCM/NOSCM. The SCM and NOSCM keywords are accepted with the same meaning as EADM/NOEADM.

REPORTS control statement

EADM|NOEADM

Specifies the Monitor I EADM Activity report.

EADM - Extended Asynchronous Data Mover (EADM) Activity Report

The Extended Asynchronous Data Mover (EADM) activity report can be used to investigate performance problems that are related to the extended asynchronous data mover facility.

How to request this report

To request the EADM Activity Report, select **3** from the Primary Menu, then select **15** from the Resource Report Selection Menu or enter the following command:

```
EADM
```

Note: This report was formerly called SCM Activity Report. The **SCM** command is still accepted and has the same meaning as **EADM**.

Contents of the report

The extended asynchronous data mover (EADM) activity report provides these types of information:

EADM level information

The EADM (extended asynchronous data mover) summary section at the top of the report provides the rate of start subchannel (SSCH) instructions for all EADM devices together with response time statistics consisting of pending, IOP queue and initial command response time.

The values cover Storage Class Memory (SCM) activity as well as EADM compression and decompression activity.

Furthermore, the section provides request rates, throughput, and ratios of compression and decompression.

The values related to compression and decompression cover asynchronous compression/decompression activity by EADM. Synchronous compression/decompression activity is not reported.

Flash Express card level information

For each Flash Express card, the report provides measurements at both the LPAR and CPC level. The rate at which internal requests are processed by the adapter card, the rate at which data units were read and written, the average response and IOP queue time is displayed.

Note: If the hardware supports Virtual Flash Memory, Flash Express cards are simulated by cache and SCM activity is reported in one report line.

Figure 2 on page 20 shows an example of the EADM Activity Report.

Command ==>		RMF 3.1 EADM Activity		Line 1 of 1	
				Scroll ==> CSR	
Samples: 60	System: SYSF	Date: 04/25/2023	Time: 13.54.00	Range: 60	Sec
----- EADM Summary -----					
SSCH Total	SSCH Rate	PEND Time	IOPQ Time	ICMR Time	
0	0.00	0.000	0.000	0.000	
Compress: Rate	Throughput	Ratio	Decompress: Rate	Throughput	Ratio
12.30	65321	45.78	23.17	43216	0.67
Card ID	Util(%)	Read(B/s)	Write(B/s)	Req Rate	Resp Time
	Part Total	Part Total	Part Total	Part Total	Part Total
VFM	0.00 0.00	0.00 0.00	0.00 0.00	0.00 0.00	0.000 0.000
					0.000

Figure 2. EADM Activity Report

Table 2. Fields in the EADM Activity Report	
Field heading	Meaning
EADM summary This section provides summary information about the extended asynchronous data mover (EADM) devices or subchannels. EADM subchannels are similar to I/O subchannels in a way that I/O instructions can be issued. However, they do not have channel paths or device numbers assigned, and they are not defined in the I/O configuration. They are created automatically during IPL.	
SSCH Total	The total number of SSCH instructions to all EADM devices in the report interval.
SSCH Rate	The number of SSCH instructions to all EADM devices per second.
PEND Time	The average function pending time across all EADM devices in milliseconds. This is similar to function pending time for traditional I/O devices, which is the amount of time between when the SSCH is issued and the first command in the channel program is accepted. $\text{PEND} = \frac{\text{Sum(Function Pending Time)}}{\text{Measurement Event Count}}$
IOPQ Time	The average IOP queue time across all EADM devices in milliseconds. This is unique to EADM devices. It represents the amount of time the request is not accepted by the adapter because it would exceed its maximum capacity. For a particular I/O request, this may occur multiple times. $\text{IOPQ} = \frac{\text{Sum(IOP Queue Time)}}{\text{Measurement Event Count}}$
ICMR Time	The average initial command response time across all EADM devices in milliseconds. This is the time from when the first command does not immediately proceed to execute until the successful start of execution at the SCM resource part. $\text{ICMR} = \frac{\text{Sum(Initial Command Response Time)}}{\text{Measurement Event Count}}$
Compress: Rate	The number of compression requests per second.
Compress: Throughput	The number of bytes compressed per second.

Table 2. Fields in the EADM Activity Report (continued)	
Field heading	Meaning
Compress: Ratio	The ratio between input bytes compressed and output bytes compressed within this interval.
Decompress: Rate	The number of decompression requests per second.
Decompress: Throughput	The number of bytes decompressed per second.
Decompress: Ratio	The ratio between input bytes decompressed and output bytes decompressed within this interval.
Flash adapter measurements	
Card ID	The identifier of the flash adapter card. <i>VFM</i> is reported if the hardware has configured Virtual Flash Memory.
Following fields are displayed at a system-wide level (Total) and for the current LPAR (Part) whereby IOPQ Time is only available at the total level.	
Util(%)	The average utilization of the flash card during the interval as reported by the SCM measurement facility. The average utilization of Virtual Flash Memory is reported as the percentage of the time spent on System Assist Processors (SAP) for SCM processing compared to the total available SAP time in this reporting interval.
Read(B/s)	Bytes read per second.
Write(B/s)	Bytes written per second.
Req Rate	The requests processed per second.
Resp Time	The average response time per request in milliseconds. The response time represents the CHSC execution time and does not include pending, IOP queue and initial command response time.
IOPQ Time	The average IOP queue time per request in milliseconds.

EADM - EADM Activity Report

The EADM Activity Report provides statistics and performance measurements on Extended Asynchronous Data Mover (EADM) activity. EADM activity encompasses Storage Class Memory (SCM) and EADM compression activity.

How to request this report

If the currently active SMFPRMxx parameter settings indicate that SMF record type 74 subtype 10 is to be collected, then RMF Monitor III gathers the data required for the EADM Activity Report into this SMF record.

To produce this report, specify

```
REPORTS(EADM)
```

Note: This report was formerly called SCM Activity Report. The SCM keyword is still accepted and has the same meaning as EADM.

This single-system report is only available in XML output format. Therefore, you need to specify the XPRPTS ddname in your Postprocessor job. *How to work with Postprocessor XML reports in z/OS Resource Measurement Facility User's Guide* provides all required information on how to produce and view XML reports.

Example URL for the DDS API

```
http://ddshost:8803/gpm/rmfpp.xml?reports=EADM
```

Contents of the report

The Extended Asynchronous Data Mover (EADM) Activity Report consists of three segments:

1. Device/subchannel level information. The EADM device summary segment provides the rate of start subchannel (SSCH) instructions for all EADM devices together with response time statistics consisting of pending, IOP queue and initial command response time.
2. Compression activity information. This segment provides request rates, throughput, and ratios of EADM compression and decompression. These values cover asynchronous compression and decompression activity by EADM. Synchronous compression and decompression is not reported.
3. Storage Class Memory activity information. For each Flash Express adapter, the report provides measurements at both the LPAR and CPC level. The total number of requests, the rate at which requests are processed by the adapter, the rate at which data units were read and written, the average response and IOP queue time is displayed.

Note: If the hardware supports Virtual Flash Memory, Flash Express cards are simulated by cache, and SCM activity is reported in one report line.

RMF Postprocessor Interval Report [System SYSE] : Extended Asynchronous Data Mover Activity Report													
RMF Version: z/OS 3.1 SMF Data: z/OS 3.1 Start : 09/30/2021-15.44.34 End : 09/30/2021-15.59.33 Interval : 15:00:00 minutes													
Device/Subchannel Summary													
Total Number of SSCH : 0 SSCH Rate : 0.00 Avg Function Pending Time : 0.000 Avg Initial Cmd Response time : 0.000													
Compression Activity													
Compression Request Rate : 12.30 Compression Throughput : 65321 Compression Ratio : 45.78 Decompression Request Rate : 23.17 Decompression Throughput : 43216 Decompression Ratio : 0.67													
Storage Class Memory Activity													
Card ID	Util% (LPAR)	Util% (Total)	Read B/Sec (LPAR)	Read B/Sec (Total)	Write B/Sec (LPAR)	Write B/Sec (Total)	Request Rate (LPAR)	Request Rate (Total)	Avg Response Time (LPAR)	Avg Response Time (Total)	Avg IOP Queue Time (Total)	Requests (LPAR)	Requests (Total)
VFM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000	0.000	0.000	0	0

Figure 3. Extended Asynchronous Data Mover (EADM) Activity Report

Table 3. Fields in the EADM Activity Report	
Field Heading	Meaning
Device/Subchannel Summary This section provides summary information about the extended asynchronous data mover (EADM) devices or subchannels. EADM subchannels are similar to I/O subchannels in a way that I/O instructions can be issued. However, they do not have channel paths or device numbers assigned, and they are not defined in the I/O configuration. They are created automatically during IPL.	
Total Number of SSCH	The total number of SSCH instructions to all EADM devices in the report interval.
SSCH Rate	The number of SSCH instructions to all EADM devices per second.
Avg Function Pending Time	The average function pending time across all EADM devices in milliseconds. This is similar to function pending time for traditional I/O devices, which is the amount of time between when the SSCH is issued and the first command in the channel program is accepted. <div> $\text{AVG} = \frac{\text{Sum(Function Pending Time)}}{\text{Measurement Event Count}}$ </div>

Table 3. Fields in the EADM Activity Report (continued)	
Field Heading	Meaning
Avg IOP Queue Time	<p>The average IOP queue time across all EADM devices in milliseconds. This is unique to EADM devices. It represents the amount of time the request is not accepted by the adapter because it would exceed its maximum capacity. For a particular I/O request, this may occur multiple times.</p> $\text{AVG} = \frac{\text{Sum(IOP Queue Time)}}{\text{Measurement Event Count}}$
Avg Initial Cmd Response Time	<p>The average initial command response time across all EADM devices in milliseconds. This is the time from when the first command does not immediately proceed to execute until the successful start of execution at the EADM resource part.</p> $\text{AVG} = \frac{\text{Sum(Initial Command Response Time)}}{\text{Measurement Event Count}}$
Compression Activity	
Compression Request Rate	The number of compression requests per second.
Compression Throughput	The number of megabytes compressed per second.
Compression Ratio	The ratio between input and output bytes compressed within this interval.
Decompression Request Rate	The number of decompression requests per second.
Decompression Throughput	The number of megabytes decompressed per second.
Decompression Ratio	The ratio between input and output bytes decompressed within this interval.
Storage Class Memory Activity	
Card ID	The identifier of the flash adapter card. VFM is reported if the hardware has configured Virtual Flash Memory.
Following fields are displayed at a system-wide level (Total) and for the current LPAR whereby IOP Queue Time is only available at the total level.	
Util(%)	<p>The average utilization of the flash card during the interval as reported by the EADM measurement facility.</p> <p>The average utilization of Virtual Flash Memory is reported as the percentage of the time spent on System Assist Processors (SAP) for EADM processing compared to the total available SAP time in this reporting interval.</p>
Read(B/Sec)	Bytes read per second.
Write(B/Sec)	Bytes written per second.
Request Rate	The requests processed per second.
Requests	The total number of requests.
Avg Response Time	The average response time per request in milliseconds. The response time represents the CHSC execution time and does not include pending, IOP queue and initial command response time.
Avg IOP Queue Time	The average IOP queue time per request in milliseconds.

SMF records

Topics that follow describe the changes to SMF records.

Type 74 Subtype 10 – Extended asynchronous data mover (EADM) statistics

This record subtype is written by the RMF Monitor III data gatherer for all extended asynchronous data mover (EADM) resources.

Extended asynchronous data mover (EADM) device (subchannel) information section

There is one section per record. This section holds the response time information as well as request rates, throughput, and ratios of compression and decompression of all EADM devices at an aggregated level.

Offsets	Name	Length	Format	Description
0	0 R7410DI	32	Structure	
0	0 R7410DSCT	4	binary	SSCH count across all devices.
4	4 R7410DNUM	4	binary	Number of updates to the time accumulation fields.
8	8 R7410DFPT	8	binary	Sum of function pending times across all devices in units of 128 microseconds. The time lapse between the SSCH being issued and the acceptance of the first command of the channel program at the device.
16	10 R7410DIQT	8	binary	Sum of IOP queue times across all devices in units of 128 microseconds. The amount of time the request is not accepted at the SCM resource because it would exceed its maximum capacity.
24	18 R7410DCRT	8	binary	Sum of initial command response times across all devices in units of 128 microseconds. The time from when the first command does not immediately proceed to execute until the successful start of execution at the SCM resource part.
32	20 R7410DFLG	1	binary	Device information flags. Bit Meaning when set 0 EADM compression facility is available. 1 - 7 Reserved.
33	21 *	7	binary	Reserved.
40	28 R7410DOCC	4	binary	Number of compression operations.
44	2C R7410DOCD	4	binary	Number of decompression operations.
48	30 R7410DISC	8	binary	Number of 1 MB input blocks consumed for compression.
56	38 R7410DOSC	8	binary	Number of 1 MB output blocks consumed for compression.
64	40 R7410DISD	8	binary	Number of 1 MB input blocks consumed for decompression.
72	48 R7410DOSD	8	binary	Number of 1 MB output blocks consumed for decompression.

I/O Queuing Activity - SMF record type 78-3**amg**

An eight-digit hexadecimal number that identifies a system alias management group.

This qualifier is required for the SuperPAV (IOXxxxxx) conditions.

chpid

A two-digit hexadecimal number that identifies the channel path ID (CHPID).

iopid

A two-digit hexadecimal number that identifies the I/O processor (IOP).

This qualifier is optional. If specified, RMF performs the exception checking just for the specified IOP identifier. If omitted, RMF loops through all IOP sections and uses the sum values for checking. In this case, the exception refers to the system wide value.

lcuid

A four-digit hexadecimal number that identifies a logical control unit.

This qualifier is required for the First-transfer-ready-disabled inhibited ratio (IOTMDINH) condition.

For all other conditions, this qualifier is optional. If it is omitted, the threshold applies to all logical control units in the SMF record.

Qualifier IOCHPID is valid only in combination with IOAMG or IOLCU.

For the condition First-transfer-ready-disabled inhibited ratio (IOTMDINH), qualifiers IOLCU and IOCHPID are required, using in the following syntax:

```
(IOLCU(lcuid),IOCHPID(chpid))
```

Examples:

```
OVERVIEW(REPORT,RECORD)
OVW(IOTMDINH(IOTMDINH(IOLCU(0027),IOCHPID(30))))

OVW(IOCUB(IOCUB(IOAMG(00000002),IOCHPID(30))))
OVW(IOCUB(IOCUB(IOAMG(00000002))))

OVW(IOXSAREQ(IOXSAREQ(IOAMG(00000002))))
```

In the **Algorithm** column:

MAX

Applies to exception operator GE, and specifies the sum of each channel path taken, where i represents channel path 0 to channel path 7.

MIN

Applies to exception operator LE, and specifies the sum of each channel path taken, where i represents channel path 0 to channel path 7.

Table 4. I/O Queuing Activity - Conditions Based on SMF Record Type 78-3				
Condition	Condition name	Qualifier	Source	Algorithm
First-transfer-ready-disabled inhibited ratio	IOTMDINH	IOLCU(lcuid) IOCHPID(chpid)	R783CTMW R783CTRD	(R783CTMW-R783CTRD) / R783CTMW
I/O processor (IOP) queue activity rate	IOPAC	iopid	R783IQCT SMF78INT	IQCT/INT
I/O processor (IOP) initiative queue average queue length	IOPQL	iopid	R783IQCT R783IQSM	(IQSM - IQCT) / IQCT
Contention rate	IOCTR	lcuid IOAMG(amg)	R783QCT SMF78INT	QCT/INT
Average queue length of delayed I/O requests	IODLQ	lcuid IOAMG(amg)	R783QCT R783QSM	(QSM-QCT)/QCT
Channel path taken rate	IOART	lcuid IOAMG(amg) IOCHPID(chpid)	R783PT SMF78INT	(PTi)/INT
Percentage of requests caused by control unit busy	IOCUB	lcuid IOAMG(amg) IOCHPID(chpid)	R783DPB R783CUB R783PT	MAX (CUBi*100)/ (PT+CUB+DPB)i MIN (CUBi*100)/ (PT+CUB+DPB)i When IOCHPID selected: CUBi*100/ (PT+CUB+DPB)i
Percentage of requests caused by director port busy	IODPB	lcuid IOAMG(amg) IOCHPID(chpid)	R783DPB R783CUB R783PT	MAX (DPBi*100)/ (PT+DPB+CUB)i MIN (DPBi*100)/ (PT+DPB+CUB)i When IOCHPID selected: DPBi*100/ (PT+CUB+DPB)i
Percent I/O processor busy	IOIPB	iopid	R783IIPB R783IPI	(IIPB * 100) / (IIPB + IPI)

Table 4. I/O Queuing Activity - Conditions Based on SMF Record Type 78-3 (continued)

Condition	Condition name	Qualifier	Source	Algorithm
Percent I/O processor busy with EADM compression / decompression	IOPECB	iopid	R783IECB R783IIPB R783IIPB	$(IECB * 100) / (IIPB + IIPB)$
Percent I/O processor busy with SCM	IOPSCB	iopid	R783ISCB R783IIPB R783IIPB	$(ISCB * 100) / (IIPB + IIPB)$
Percent I/O processor idle	IOPIPI	iopid	R783IIPB R783IIPB	$(IIPB * 100) / (IIPB + IIPB)$
Rate I/O functions started	IORIFS	iopid	R783IIFS SMF78INT	IIFS / INT
Rate processed I/O interrupts	IORPII	iopid	R783IPII SMF78INT	IPII / INT
Percent of I/O retries	IOPALB	iopid	R783ICHB R783IDPB R783ICUB R783IDVB R783IIFS	$((\text{Sum all retries}) * 100) / (IIFS + \text{Sum all retries})$
Percent of I/O retries due to channel busy	IOPCHB	iopid	R783ICHB R783IDPB R783ICUB R783IDVB R783IIFS	$(ICHB * 100) / (IIFS + \text{Sum all retries})$
Percent of I/O retries due to director port busy	IOPDPB	iopid	R783ICHB R783IDPB R783ICUB R783IDVB R783IIFS	$(IDPB * 100) / (IIFS + \text{Sum all retries})$
Percent of I/O retries due to control unit busy	IOPCUB	iopid	R783ICHB R783IDPB R783ICUB R783IDVB R783IIFS	$(ICUB * 100) / (IIFS + \text{Sum all retries})$
Percent of I/O retries due to device busy	IOPDVB	iopid	R783ICHB R783IDPB R783ICUB R783IDVB R783IIFS	$(IDVB * 100) / (IIFS + \text{Sum all retries})$
Number of I/O retries per SSCH	IONALB	iopid	R783ICHB R783IDPB R783ICUB R783IDVB R783IIFS	$(\text{Sum all retries}) / IIFS$
Number of I/O retries per SSCH due to channel busy	IONCHB	iopid	R783ICHB R783IIFS	ICHB / IIFS
Number of I/O retries per SSCH due to director port busy	IONDPB	iopid	R783IDPB R783IIFS	IDPB / IIFS
Number of I/O retries per SSCH due to control unit busy	IONCUB	iopid	R783ICUB R783IIFS	ICUB / IIFS
Number of I/O retries per SSCH due to device busy	IONDVB	iopid	R783IDVB R783IIFS	IDVB / IIFS
Average control unit busy delay time	IOCBT	lcuid IOAMG(amg) IOCHPID(chpid)	R783CBT R783PT	$\text{Sum}(\text{CBT}) / \text{Sum}(\text{PT})$

Table 4. I/O Queuing Activity - Conditions Based on SMF Record Type 78-3 (continued)				
Condition	Condition name	Qualifier	Source	Algorithm
Average initial command response time	IOCMR	lcuid IOAMG(amg) IOCHPID(chpid)	R783CMR R783PT	Sum(CMR)/Sum(PT)
Average channel subsystem delay time	IOCSS	lcuid IOAMG(amg)	R783CSST R783PT	CSST/Sum(PT)
HyperPAV wait ratio	IOHWAIT	lcuid IOAMG(amg)	R783HNAI R783HTIO	HNAI/HTIO When IOAMG selected: (Sum(HNAI))/ (Sum(HTIO))
Maximum number of in-use HyperPAV aliases	IOHMAX	lcuid IOAMG(amg)	R783HAIU R783XHBC	HAIU+XHBC When IOAMG selected: Maximum over all LCUs of that AMG MAX(HAIU+XHBC)
Maximum number of in-use HyperPAV aliases for one device	IOHDMAX	lcuid IOAMG(amg)	R783HCAD	HCAD When IOAMG selected: Maximum over all LCUs of that AMG
The high watermark of queued I/O requests	IOHIOQC	lcuid IOAMG(amg)	R783HIOQ	HIOQ When IOAMG selected: Maximum over all LCUs of that AMG
Ratio of successful alias requests	IOXSAREQ	IOAMG(amg)	R783XAUC R783XANC	Sum(XAUC)/ Sum(XANC)
Ratio of unsuccessful alias requests in home LCU	IOXUAHRQ	IOAMG(amg)	R783XNHC R783XANC	Sum(XNHC)/ Sum(XANC)
Rate of aliases borrowed from peer LCUs	IOXABC	IOAMG(amg)	R783XABC SMF78INT	Sum(XABC)/INT
High water mark of concurrently borrowed aliases	IOXHCBA	IOAMG(amg)	R783XHBC	XHBC
Rate of aliases loaned to a peer LCU	IOXALC	IOAMG(amg)	R783XALC SMF78INT	Sum(XALC)/INT
High water mark of concurrently loaned aliases to a peer LCU	IOXHCLA	IOAMG(amg)	R783XHLC	XHLC
Average queue length when an alias was needed	IOXCQD	IOAMG(amg)	R783XCQD R783XANC	Sum(XCQD)/ Sum(XANC)
Average number of in-use aliases when an alias was needed	IOXIUAC	IOAMG(amg)	R783XCIU R783XANC	Sum(XCIU)/ Sum(XANC)

Type 78 Subtype 3 I/O Queuing Activity

I/O queuing global section

This section contains triplet SMF78QDS.

Offsets	Name	Length	Format	Description
0	0 R783GFLG	1	binary	IOQ global flags Bit Meaning when set 0 Incorrect data because channel measurement facility failed 1 DIAGNOSE interface failed 2 Store Primary Queue Data not supported 3 DCM supported by hardware 4 Configuration contains DCM managed channels 5 IOP utilization data supported 6 Initial command response time measurements supported 7 First-transfer-ready-disabled data available
1	1 R783GFLX	1	binary	IOQ global flags extended Bit Meaning when set 0 Alias management groups available. 1 EADM compression facility available. 2 Storage-class memory measurement facility available. 3 - 7 Reserved.
2	2 R783GNTR	2	binary	Number of descriptor triplets following.
4	4 R783GIDS	4	binary	Offset to I/O Processor (IOP) Initiative Queue data section.
8	8 R783GIDL	2	binary	Length of I/O Processor (IOP) Initiative Queue data section.
10	A R783GIDN	2	binary	Number of I/O Processor (IOP) Initiative Queue data sections.
12	C	4		Reserved.
16	10 R783TSR	2	binary	Total number of small records written during interval.
18	12	2		Reserved.
20	14 R783TOT	4	binary	Total number of data sections recorded during the interval.
24	18 R783NXT	4	binary	Total number of data sections in the following record.

Offsets	Name	Length	Format	Description
28	1C R783CFL	1	binary	Configuration change flags Bit Meaning when set 0 Configuration changed. Used to decide whether to provide the text “POR” or “ACTIVATE” on reports. Also used to check whether data can be combined in a duration report. 1 Configuration change since power on reset (POR). 2 POR using IOC data set that contains a token. 3 I/O token is valid. 4 Hardware allows multiple channel subsystems. 5-7 Reserved.
29	1D R783CSS	1	binary	Channel Subsystem ID. Only valid if bit 4 of R783CFL is set.
30	1E	2		Reserved.
32	20 R783TNM	44	EBCDIC	IODF name.
76	4C R783TSF	2	EBCDIC	IODF name suffix.
78	4E	2		Reserved.
80	50 R783TOK	16	EBCDIC	Partial token information.
80	50 R783TDT	8	EBCDIC	IODF creation date, in the form <i>mm/dd/yy</i> .
88	58 R783TTM	8	EBCDIC	IODF creation time, in the form <i>hh.mm.ss</i> .
96	60 R783TDY	10	EBCDIC	IODF creation date, in the form <i>mm/dd/yyyy</i> .
106	6A	2		Reserved.

IOP Initiative Queue and Utilization Data Section

This section contains one entry per IOP described by triplet R783GIDS. The contents of the fields R783IIPB through R783IDVB are valid if bit 5 of R783GFLG is set.

Offsets	Name	Length	Format	Description
0	0 R783IQID	2	binary	Input output processor (IOP) initiative queue identifier.
2	2 R783IFLG	1	binary	Input output processor (IOP) Flags Bit Meaning when set 0 Input output processor (IOP) is installed. 1-7 Reserved.
3	3 *	1	*	Reserved.
4	4 R783IQSM	4	binary	Accumulator is incremented by the current queue length in the Input output processor (IOP) whenever a request is enqueued.
8	8 R783IQCT	4	binary	Number of elements enqueued on the Input output processor (IOP) initiative queue.
12	C *	4	*	Reserved.
16	10 R783IIPB	8	floating	Number of times the I/O processor was busy.
24	18 R783IIPB	8	floating	Number of times the I/O processor was idle.

Offsets	Name	Length	Format	Description
32 20	R783IIFS	8	floating	Number of I/O functions initially started.
40 28	R783IPII	8	floating	Number of processed I/O interrupts.
48 30	R783ICPB	8	floating	Number of times an I/O was retried due to channel path busy.
56 38	R783IDPB	8	floating	Number of times an I/O was retried due to director port busy.
64 40	R783ICUB	8	floating	Number of times an I/O was retried due to control unit busy.
72 48	R783IDVB	8	floating	Number of times an I/O was retried due to device busy.
80 50	R783ISCB	8	floating	Number of times the I/O processor was busy with SCM operations.
88 58	R783IECB	8	floating	Number of times the I/O processor was busy with compression or decompression.
96 60	*	8	*	Reserved.

IOP Initiative Queue and Utilization Data Section

This section contains one entry per IOP described by triplet R783GIDS. The contents of the fields R783IIPB through R783IDVB are valid if bit 5 of R783GFLG is set.

Offsets	Name	Length	Format	Description
0 0	R783IQID	2	binary	Input output processor (IOP) initiative queue identifier.
2 2	R783IFLG	1	binary	Input output processor (IOP) Flags Bit Meaning when set 0 Input output processor (IOP) is installed. 1-7 Reserved.
3 3	*	1	*	Reserved.
4 4	R783IQSM	4	binary	Accumulator is incremented by the current queue length in the Input output processor (IOP) whenever a request is enqueued.
8 8	R783IQCT	4	binary	Number of elements enqueued on the Input output processor (IOP) initiative queue.
12 C	*	4	*	Reserved.
16 10	R783IIPB	8	floating	Number of times the I/O processor was busy.
24 18	R783IIPB	8	floating	Number of times the I/O processor was idle.
32 20	R783IIFS	8	floating	Number of I/O functions initially started.
40 28	R783IPII	8	floating	Number of processed I/O interrupts.
48 30	R783ICPB	8	floating	Number of times an I/O was retried due to channel path busy.
56 38	R783IDPB	8	floating	Number of times an I/O was retried due to director port busy.
64 40	R783ICUB	8	floating	Number of times an I/O was retried due to control unit busy.
72 48	R783IDVB	8	floating	Number of times an I/O was retried due to device busy.
80 50	R783ISCB	8	floating	Number of times the I/O processor was busy with SCM operations.
88 58	R783IECB	8	floating	Number of times the I/O processor was busy with compression or decompression.
96 60	*	8	*	Reserved.

Chapter 4. z/OS File System data

Compress data in the z/OS File System (zFS) to save disk space.

You can compress new data and existing data.

To compress new data:

- Set a global default used by all formatting methods when formatting a new file system, with an option in the IOEFSPRM configuration file, `format_compression=on`. The default for this option is to set compression off. Alternatively, if you leave the global default for compression off, you can explicitly request compression on a formatting method with the `-compress` keyword.
- Explicitly specify compression on a formatting method even if the global default is that compression is off, with the `-compress` keyword.

To compress existing data use the `zfsadm compress` command.

To monitor the compression status use the `zfsadm fsinfo` command.

Assessing compression: Assess the compression by comparing the size of the file system with compression to the size of the file system before compression. For performance metrics, use:

- SMF record type 30.
- RMF EADM reporting (RMF 74.10).

For more information, see [“Assess compression of zFS data”](#) on page 34.

The compression process

The compression process uses zEDC. The average amount of disk space that is saved per file averages approximately 65%, depending on the type of data that is being compressed.

If you cancel a compression that is in progress, the zFS file system will remain partially compressed. In a partially compressed file system, new files may or may not be compressed. You can resume the compression later with another **zfsadm compress** command.

The compression process is not mandatory. If the compression of a file does not reduce space, the file is left in its uncompressed format.

Important: As soon as a compressed file system is seen by zFS, the `edcfixd` option is applied to the cache even if the option was not specified in IOEFSPRM.

Restrictions:

1. Do not enable compression for any file system until you migrate all of your systems to z/OS V2R3. All systems in a sysplex must be at least z/OS V2R3 before any file systems are compressed because compression is not supported prior to z/OS V2R3. Also, do not use compression until you know that no system will be regressed to a prior release. Compressed file systems cannot be mounted on a release prior to V2R3. Therefore, if there is no zFS system in the shared file system environment that is eligible to own a compressed file system, the file system will be inaccessible.

Decompression is supported if there are pre-V2R3 systems in the sysplex in order to allow the compression to be backed out.

2. Only files larger than 8 K can be compressed. Directories and other control information inside the zFS file system are not compressed.
3. You cannot compress or decompress an aggregate that is in a partially encrypted or partially decrypted state. In other words, if an encryption or decryption process was stopped for an aggregate, you cannot compress or decompress that aggregate until after the encryption or decryption is completed.

Defining a new file system that is always compressed

The IOEFSPRM configuration option `format_compression=on` indicates a global default that is used by all formatting methods when determining the default compression behavior while formatting a new file system. This global compression default can be overridden by specifying the `-nocompress` keyword.

If IOEFSPRM configuration option `format_compression=off` is specified, all formatting methods can explicitly specify the `-compress` keyword to format the file system with compression.

The following example is JCL for defining and compressing a new aggregate.

```
//ZDEFFMT JOB , 'DEF FORMAT COMPRESS',
//          MSGCLASS=H,
//          CLASS=A,
//          TIME=(1440),MSGLEVEL=(1,1)
//*-----
//*  DEFINE FORMAT COMPRESS
//*-----
//*
//DEFINE    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//AMSDUMP  DD SYSOUT=H
//DASD0    DD DISP=OLD,UNIT=3390,VOL=SER=SMBRS3
//SYSIN     DD *
//          DEFINE CLUSTER (NAME(SUIMGNS.HIGHRISK.TEST) -
//          ZFS CYL(2 0) SHAREOPTIONS(3))
//*
//CREATE    EXEC PGM=IOEFSUTL,REGION=0M,
//          PARM=('format -aggregate SUIMGNS.COMPRESS.TEST -compress')
//SYSPRINT DD SYSOUT=H
//STDOUT    DD SYSOUT=H
//STDERR    DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//CEEDUMP  DD SYSOUT=H
```

The following example uses the **zfsadm format** command with the `-compress` option to compress the new file system.

```
zfsadm format -aggregate PLEX.DCEIMGNJ.ENC -compress
IOEZ00077I HFS-compatibility aggregate PLEX.DCEIMGNJ.ENC was successfully created.
```

zfsadm compress

This is an excerpt from the topic. For the complete topic, see [zfsadm compress](#) in *z/OS File System Administration*.

Purpose

zfsadm compress compresses a zFS aggregate.

Format

```
zfsadm compress -aggregate name [-cancel][-trace file_name][-level][-help]
```

Options

-aggregate *name*

Specifies the name of the aggregate to be compressed. The aggregate name is not case-sensitive. It is always converted to uppercase.

-cancel

Cancels an in-progress compress operation for the specified aggregate.

Usage notes

1. The **zfsadm compress** command is a long-running administrative command that uses the zEDC compression method to compress an existing zFS aggregate.
2. To improve performance of the compression I/O, IBM recommends that you specify the `edcfixd` option in the IOEFSPRM parameter `user_cache_size`.
3. If the user cache is not registered with the zEDC Express service, zFS will attempt to register the user cache after the **zfsadm compress** command completes. zFS constraints might prevent zFS from registering the entire user cache with the zEDC Express service. The **zfsadm compress** command will cause the ZFS_VERIFY_COMPRESSION_HEALTH check to be run.
4. To process the compression request, the long-running command thread pool must have an available foreground thread. See the IOEFSPRM configuration option `long_cmd_threads` for information about controlling the size of the long-running foreground and background thread pools.
5. The command must be issued from a z/OS V2R3 or later system, and the zFS file system must be zFS-owned on a z/OS V2R3 or later system. The aggregate must be at least aggregate version 1.5 and mounted read/write. Do not use this command before you have migrated all your systems to z/OS V2R3 or later. If there are systems that are active prior to z/OS V2R3 in the shared file system environment, compression will not take place.
6. zFS will determine whether the compression can achieve space savings. If not, it will not perform compression. Only regular files that are stored in blocked format can be compressed. Applications can still access the aggregate while it is being compressed.
7. A compress operation can be interrupted by using the `-cancel` option, UNMOUNT immediate with the `-force` option, or during a shutdown. If the compress operation is interrupted, the zFS aggregate might be left with both compressed and uncompressed files. This partial state is allowed. Another **zfsadm compress** command can be issued to resume the compression operation for the rest of the files after the interruption.
8. You cannot compress an aggregate that is in a partially encrypted or partially decrypted state. In other words, if encryption or decryption was interrupted for an aggregate, you cannot compress it.
9. Use either the **zfsadm fsinfo** or MODIFY FSINFO command to display whether an aggregate is compressed or is being compressed. Progress of the compress operation can be seen in the owner status display.
10. The **zfsadm fileinfo** command can be used to show whether a particular file is compressed or not.
11. The backup change activity flag is set if any file data is compressed.
12. Aggregates with active file backups cannot be compressed.

Examples

1. The following command compresses an existing zFS aggregate:

```
zfsadm compress -aggregate PLEX.ZFS.AGGR1
IOEZ00899I Aggregate PLEX.ZFS.AGGR1 successfully compressed.
```

Monitoring and displaying the compression status

Use the **zfsadm fsinfo** command to monitor the compression status. To display the compression status, use either **zfsadm fileinfo** or **zfsadm fsinfo**.

The following example uses **zfsadm fsinfo** to monitor the compression status.

```
zfsadm fsinfo -aggregate PLEX.DCEIMGNJ.BIGENC
File System Name: PLEX.DCEIMGNJ.BIGENC
*** owner information ***
.....
```

```
Status:          RW,RS,NE,CI
...
...
Compress Progress: running, 48% started at Nov 21 16:34:40 2016 task 57F5E0
...

Legend: RW=Read-write, RS=Mounted RWSHARE, NE=Not encrypted
        CI=Partially compressed
```

The following example uses **zfsadm fsinfo** with the **-basic** option to display the compression status.

```
zfsadm fsinfo -aggregate PLEX.DCEIMGNJ.BIGENC -basic
PLEX.DCEIMGNJ.BIGENC                                DCEIMGNJ RW,RS,EI,CO

Legend: RW=Read-write, RS=Mounted RWSHARE, EI=Partially Encrypted
        CO=Compressed
```

The following example uses **zfsadm query** with the **-compress** option to monitor the compression effectiveness and performance of zEDC services.

```
zfsadm query -compress

Compression calls:      246428  Avg. call time:          0.177
KB input               13190960  KB output              1971456
Decompression calls:   509140  Avg. call time:          0.154
KB input               4073128  KB output              21406072
```

The **zfsadm fileinfo** command shows an exact count of kilobytes saved for a file that is compressed. The following example uses **zfsadm fileinfo** to display the compression status.

```
zfsadm fileinfo /tst/myfile
path: /tst/myfile
*** global data ***
...
mtime      Nov  2 11:21:01 2015    atime      Nov  2 11:21:01 2015
ctime      Nov  2 11:21:01 2015    create time Nov  2 11:21:01 2015
reftime    none
not encrypted

compressed 4762K saved
```

Assess compression of zFS data

You can assess compression of zFS data with SMF records and RMF reports.

For performance metrics use:

- SMF record type 30. See [“Assess compression of z/OS data”](#) on page 13.
- RMF EADM reporting (RMF 74.10). See [“RMF EADM reporting”](#) on page 18.

For more information about SMF, see [z/OS MVS System Management Facilities \(SMF\)](#).

For more information about RMF reports, see [z/OS Resource Measurement Facility User's Guide](#).

Chapter 5. z/OS applications

Support for in-application compression is provided through zlib, which is an open source data compression library. The zlib compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data.

zEDC uses a modified version of the zlib compression library. The IBM-provided zlib-compatible C library provides a set of wrapper functions that use zEDC compression when appropriate and software-based compression services when zEDC is not appropriate.

If you have been using zEDC with the zEDC Express adapter, review your applications to consider applying compression more broadly. Ensure that the z/OS input buffers for the Java application are sufficient for what is listed for DEFMINREQSIZE and INFMINREQSIZE displayed with the D IQP command. You do not need to re-link the application to use the IBM-provided zlib.

If you are not already using zEDC, define applications to use compression as follows:

1. In a Java application, use the java.util.zip compression classes.
2. In a C application, use the supported zlib APIs. Link or re-link applications to use the IBM-provided zlib, which is an archive file in the z/OS UNIX System Services file system and can be statically linked into your applications.
3. Enable zEDC for an application. See [Enabling zEDC \(www.ibm.com/docs/en/sdk-java-technology/8?topic=devices-zenterprise-data-compression-zos-only\)](http://www.ibm.com/docs/en/sdk-java-technology/8?topic=devices-zenterprise-data-compression-zos-only) for more information.
4. For C applications, link or re-link the application to use the IBM-provided zlib.

Path for the zlib archive file: /usr/lpp/hzc/lib/libz.a

Path for the zlib header files: /usr/lpp/hzc/include/

Re-link an application when a new service is provided in zlib in order to use the updated zlib and take advantage of the new function. zlib will always be compatible -- Integrated Accelerator for zEDC will never break zlib's ability to call into the operating system to get accelerated compression.

Assessing compression: After your application has run, you can assess the compression using:

- SMF record type 30.
- SMF record type 113.

See [“Assess compression of z/OS applications”](#) on page 36.

Displaying PCIE-related parameters (IQP)

Use the DISPLAY IQP command to display parameters that are used for managing PCIE-related features, such as the zEDC Express feature.

```
D IQP[,L={a|name|name-a}]
```

IQP

The system issues the IQP066I message to display the IQP parameters. The IQP parameters are defined in the IQPPRMxx parmlib member. For more information about the IQPPRMxx member, see [z/OS MVS Initialization and Tuning Reference](#).

L=a, name, or name-a

Specifies the display area (*a*), console name (*name*), or both (*name-a*) where the display is to appear.

If you omit this operand, the display is presented in the first available display area or the message area of the console through which you enter the command.

Assess compression of z/OS applications

After your applications have run, you can assess compression with SMF records.

For performance metrics use:

- SMF record type 113 for synchronous processing. (zlib is always synchronous on a z15.)
- SMF record type 30 to see the change in elapsed time and CPU time for the job. See [“Assess compression of z/OS data” on page 13](#).

For more information about SMF, see [z/OS MVS System Management Facilities \(SMF\)](#).

Record type 113 (X'71') – Hardware capacity, reporting, and statistics

The system writes record type 113 to record hardware capacity, reporting, and statistics for IBM System z10 or later CPCs. With the enhanced-monitor facility released with the IBM z196 and later CPCs, the SMF record can be utilized to capture software events.

For more information about hardware data event collection, see [Setting up hardware event data collection in z/OS MVS System Commands](#).

IBM makes the following recommendations:

- Products should process SMF type 113 subtype 1 records, when available, because that is where future enhancements will be made. If subtype 1 records are not available, products may process the subtype 2 records.
- Customers should collect SMF type 113 subtype 1 and 2 records.

Record mapping

Header section

This section contains the common SMF record headers fields and the triplet fields (offset/length/number), if applicable, that locate the other sections on the record.

Offsets	Name	Length	Format	Description
0	0 SMF113LEN	2	binary	Record length.
2	2 SMF113SEG	2	binary	Segment descriptor.
4	4 SM113FLG	1	binary	Header flag byte: <div> Bit Meaning when set 0 Subsystem identification follows system identification 1 Subtypes used 2 Reserved 3-6 Version indicators 7 Reserved </div>
5	5 SMF113RTY	1	binary	Record type 113 (X'71')
6	6 SMF113TME	4	packed	Time since midnight, in hundredths of a second, when the record was moved into the SMF buffer.
10	A SMF113DTE	4	packed	Date that the record was moved into the SMF buffer, in the form <i>OcyydddF</i> .
14	E SMF113SID	4	EBCDIC	System identification (from the SID parameter).
18	12 SMF113WID	4	EBCDIC	Subsystem identifier.

Offsets	Name	Length	Format	Description
22	16 SMF113STY	2	binary	Indicates the record subtype: Subtype Description 1 Hardware event counter deltas 2 Hardware event counters
24	18 *	2	*	Reserved.
26	1A SMF113SDL	2	binary	Length of self-defining section.

Self-defining section, based on the address of SMF record type 113 + the length of the header section

Offsets	Name	Length	Format	Description
0	0 SMF113SOF	4	binary	Offset to subsystem section from beginning of record type 113
4	4 SMF113SLN	2	binary	Length of subsystem section
6	6 SMF113SON	2	binary	Number of subsystem sections
8	8 SMF113IOF	4	binary	Offset to identification section from beginning of record type 113
12	0C SMF113ILN	2	binary	Length of identification section
14	0E SMF113ION	2	binary	Number of identification sections
16	10 SMF113DOF	4	binary	Offset to data section from beginning of record type 113
20	14 SMF113DLN	2	binary	Length of data section
22	16 SMF113DON	2	binary	Number of data sections

Subsystem section, based on the address of SMF record type 113 + the offset value in SMF113SOF

This section contains information about the subsystem that generated the record.

Offsets	Name	Length	Format	Description
0	0	2		Reserved
2	2 SMF113RVN	2	EBCDIC	Record version number
4	4 SMF113PNM	8	EBCDIC	Product name
12	C SMF113OSL	8	EBCDIC	MVS product level

Identification section, based on the address of SMF record type 113 + the offset value in SMF113IOF

Offsets	Name	Length	Format	Description
0	0 SMF113JBN	8	EBCDIC	Job name
8	8 SMF113RST	4	binary	Reader start time
12	C SMF113RSD	4	packed	Reader start date
16	10 SMF113STP	8	EBCDIC	Step name
24	18 SMF113IntervalStart	8	binary	Interval start time, STCK format
32	20 SMF113IntervalEnd	8	binary	Interval end time, STCK format

Subtype 1

Subtype 1 of record type 113 contains event counter deltas for IBM System z10 or later CPCs. You can calculate its location in the SMF record using the following formula:

$$(address\ of\ record\ type\ 113 + offset\ value\ in\ field\ SMF113DOF)$$

Subtype 1 records are CPU or core specific. For each hardware data event collection cycle:

- If one or more CPU counter sets are being collected (see the MODIFY HISPROC command), the system creates one subtype 1 record for each active CPU. If one or more core counter sets are being collected (see the MODIFY HISPROC command) in addition, each core counter set will be recorded from one CPU on each core.
- If no CPU counter sets are being collected (see the MODIFY HISPROC command) and one or more core counter sets are being collected (see the MODIFY HISPROC command), the system creates one subtype 1 record for one active CPU on each core.

Each event counter represents the number of times a specific event has occurred from the start of the current hardware data event collection cycle or since the previous SMF Type 113 subtype 1 record was written. The system captures the valid counters and places them in the appropriate counter data slot, based on the counter event. If a counter data slot is zero, either the counter event is not defined or the counter event has not occurred in the current collection cycle. For example, counter event 0 is located in the 1st counter data slot, and counter event 9 is in the 10th slot.

Header/self-defining section, based on the address of SMF record type 113 + the offset value in SMF113DOF

This section contains the common SMF subtype header fields and the triplet fields (offset/length/number), if applicable, that locate the other sections on the record.

Offsets	Name	Length	Format	Description
0	0 SMF113_1_CTS	8	binary	Time when the hardware data collection run started in STCK format
8	8 SMF113_1_CTM	8	binary	Time when this SMF record was written in STCK format.
16	10 SMF113_1_CpuId	2	binary	Processor ID for which the hardware counters are recorded. Note that zero is a valid processor number.
18	12 SMF113_1_CpuProcClass	1	binary	The processor type for which the hardware event counters are recorded. Is one of the following: 0 Standard CP 2 zAAP 4 zIIP
19	13	1	binary	Reserved.
20	14 SMF113_1_CpuSpeed	4	binary	Processor speed for which the event counters are recorded. Speed is in cycles/microsecond.
24	18 SMF113_1_MachType	4	EBCDIC	The machine type.
28	1C SMF113_1_MachModel	16	EBCDIC	The machine model.
44	2C SMF113_1_CtrVersion0	2	binary	Zero counter version number. This number is increased when there is a change to the meaning of a counter in the z/OS counter set.
46	2E SMF113_1_CtrVersion1	2	binary	First counter version number. This number is increased when there is a change to the meaning of a counter or the number of installed counters in the Basic or Problem-state counter sets.

Offsets	Name	Length	Format	Description
48	30 SMF113_1_CtrVersion2	2	binary	Second counter version number. This number is increased when there is a change to the meaning of a counter or the number of installed counters in the Crypto-activity or Extended or MT-diagnostic counter sets.
50	32 SMF113_1_Flags	2	binary	Record flags: Bit Meaning when set 0 The hardware indicated the hardware has lost counter data during the current interval. 1 The hardware indicated the hardware has lost MT counter data during the current interval.
Self-defining section				
52	34 SMF113_1_CSOF	4	binary	Offset to counter set section, from beginning of r SMF record type 113.
56	38 SMF113_1_CSLN	2	binary	Length of counter set section.
58	3A SMF113_1_CSON	2	binary	Number of counter set sections.
60	3C SMF113_1_MachSeqCode	16	EBCDIC	The machine sequence code.
76	4C SMF113_1_CoreId	2	binary	Core ID for which the hardware event counters are recorded. Note that zero is a valid core ID number.

Counter set section, based on the address of SMF record type 113 + the offset value in field SMF113_1_CSOF

Offsets	Name	Length	Format	Description
0	0 SMF113_1_CSType	2	binary	Counter set type for counters recorded: 1 Basic counter set 2 Problem-state counter set 3 Crypto counter set 4 Extended counter set 5 z/OS counter set 6 MT-diagnostic counter set
2	2 SMF113_1_CSFlags	2	binary	Record flags: Bit Meaning when set 0 This counter set's counter data section is mapped by the SMF113_1_LCDS data area. Otherwise, it is mapped by the SMF113_1_SCDS data area.
4	4 SMF113_1_CDOP	4	binary	Offset to counter data section for this counter set, from the beginning of the record. This can be to a section mapped by the SMF113_1_SCDS data area or a section mapped by the SMF113_1_LCDS data area, depending on bit 0 of SMP113_1_Flags.
8	8 SMF113_1_CDLN	2	binary	Length of counter data section.
10	A SMF113_1_CDON	2	binary	Number of counter data sections.

Short counter data section, based on the address of SMF record type 113 + the offset value in field SMF113_1_CDOF

Offsets	Name	Length	Format	Description
0	0 SMF113_1_SCR	4	binary	Event counter value delta. This is the number of times a particular counter event has occurred since either the start of the HIS collection run, or the previous SMF 113 type 1 record was written.

Long counter data section, based on the address of SMF record type 113 + the offset value in field SMF113_1_CDOF

Offsets	Name	Length	Format	Description
0	0 SMF113_1_LCR	8	binary	Event counter value delta. This is the number of times a particular counter event has occurred since either the start of the HIS collection run, or the previous SMF 113 type 1 record was written.

Subtype 2

Subtype 2 of record type 113 contains hardware data event counters for IBM System z10 or later CPCs. You can calculate its location in the SMF record using the following formula:

(address of record type 113 + offset value in field SMF113DOF)

Subtype 2 records are CPU specific. For each hardware data event collection cycle, the system creates one subtype 2 for each active CPU.

Only a subset of the hardware data CPU event counters can be collected in subtype 2 records. (See the description of the SMF113_2_CST field for the supported counter sets.) IBM recommends using SMF type 113 subtype 1 records.

The system captures the valid counters and places them contiguously in subtype 2 of record type 113. For example, counter 0 of the first counter set, will be in the 1st slot. Counter 9 of the first counter set will be in slot 10 of the first counter set. Counter 0 of the second counter set will follow the last counter in the first counter set.

Header/self-defining section, based on the address of SMF record type 113 + the offset value in SMF113DOF

This section contains the common SMF subtype header fields and the triplet fields (offset/length/number), if applicable, that locate the other sections on the record.

Offsets	Name	Length	Format	Description
0	0 SMF113_2_CTS	8	binary	Time when the hardware data collection run started in STCK format
8	8 SMF113_2_CTM	8	binary	Time when this SMF record was written in STCK format.
16	10 SMF113_2_CPU#	1	binary	This field is deprecated, use SMF113_2_CpuId instead. Processor number for which the hardware counters in SMF113_2_CR are recorded. Note that zero is a valid processor number.
17	11 SMF113_2_CpuProcClass	1	binary	The processor type for which the hardware event counters are recorded. Is one of the following: 0 Standard CP 2 zAAP 4 zIIP

Offsets	Name	Length	Format	Description
18 12	SMF113_2_CF	2	binary	<p>Record flags:</p> <p>Bit</p> <p>Meaning when set</p> <p>0</p> <p>First SMF record for the hardware data collection run. The counter values are the initial values at the beginning of the run.</p> <p>1</p> <p>Intermediate SMF record, written by the system at defined intervals during the hardware data collection run. The counter values are intermediate values. The interval is based on the SMFINTVAL parameter specified at the start of the hardware data collection run.</p> <p>2</p> <p>Final SMF record written for this hardware data collection run. The counter values are the final values.</p> <p>3</p> <p>Indicates that the SMF record was written on non-standard hardware.</p> <p>4</p> <p>When ON, the hardware indicated the hardware has lost counter data during the current interval.</p> <p>5-15</p> <p>Reserved.</p>
20 14	SMF113_2_CTRVN1	2	binary	First counter version number. This number is increased when there is a change to the meaning of a counter or a change to the number of the installed counters in the basic or problem-state counter sets.
22 16	SMF113_2_CTRVN2	2	binary	Second counter version number. This number is increased when there is a change to the meaning of a counter or a change to the number of the installed counters in the crypto-activity or extended counter sets.
Self-defining section				
24 18	SMF113_2_CSOF	4	binary	Offset to counter set section, from beginning of SMF record type 113
28 1C	SMF113_2_CSLN	2	binary	Length of counter set sections
30 1E	SMF113_2_CSON	2	binary	Number of counter set sections
32 20	SMF113_2_CDOF	4	binary	Offset to counters section, from beginning of SMF record type 113
36 24	SMF113_2_CDLN	2	binary	Length of counters sections
38 26	SMF113_2_CDON	2	binary	Number of counter sections
CPU information section				
40 28	SMF113_2_CPSP	4	binary	Processor speed for which the hardware event counters are recorded. Speed is in cycles/microsecond.
44 2C	SMF113_2_MachType	4	EBCDIC	The machine type.
48 30	SMF113_2_MachModel	16	EBCDIC	The machine model.
64 40	SMF113_2_CpuId	2	binary	<p>Processor ID for which the hardware event counters are recorded.</p> <p>Note that zero is a valid processor number.</p>
66 42	*	2	binary	Reserved
68 44	SMF113_2_MachSeqCode	16	EBCDIC	The machine sequence code.

Record type 113

Note: For more information on machine type and machine model, see [Resource Link home page](http://www.ibm.com/servers/resourcelink) (www.ibm.com/servers/resourcelink).

Counter set section, based on the address of SMF record type 113 + the offset value in field SMF113_2_CSOF

Offsets	Name	Length	Format	Description
0	0 SMF113_2_CST	1	binary	Indicates the counter set type for counters recorded in field SMF113_2_CR: 1 Basic counter set 2 Problem-state counter set 3 Crypto counter set 4 Extended counter set
1	1	1		Reserved
2	2 SMF113_2_CSN	2	binary	Number of counter sections.
4	4	8		Reserved

Counters section, based on the address of SMF record type 113 + the offset value in field SMF113_2_CDOF

Offsets	Name	Length	Format	Description
0	0 SMF113_2_CR	8	binary	Hardware event counter value. This is the absolute number of times a particular hardware counter event has occurred.

Chapter 6. z/OS databases

Compress Db2 large object (LOB) table spaces, archive logs, and IBM Content Manager OnDemand archives to achieve faster query response times, reduced disk use, and improved archive performance.

Compressing LOB data with zEDC can reduce the size of data in a LOB table space and ensure that the LOB table space and its associated base table space remain available for longer. For more information, see [COMPRESS DIRLOB \(www.ibm.com/docs/en/db2-for-zos/12?topic=ddcdp-compress-db2-dir-lob-field-compress-dirlob-subsystem-parameter\)](http://www.ibm.com/docs/en/db2-for-zos/12?topic=ddcdp-compress-db2-dir-lob-field-compress-dirlob-subsystem-parameter).

If you have been using zEDC with the zEDC Express adapter, review the definitions, including ARS.CFG parameters for Content Manager OnDemand, and consider applying compression more broadly. For more information on ARS.CFG parameters, see [ARS.CFG parameters \(www.ibm.com/docs/en/cmofz/10.1.0?topic=file-zedc-parameters\)](http://www.ibm.com/docs/en/cmofz/10.1.0?topic=file-zedc-parameters).

If you have not been using zEDC, enable compression for individual table spaces by specifying the COMPRESS YES option on the CREATE TABLESPACE or ALTER TABLESPACE statement. For more information, see [CREATE TABLESPACE \(www.ibm.com/docs/en/db2-for-zos/12?topic=statements-create-tablespace\)](http://www.ibm.com/docs/en/db2-for-zos/12?topic=statements-create-tablespace). Enable compression for the Db2 directory by setting the COMPRESS_DIRLOB subsystem parameter to YES.

Assessing compression: After testing the Db2 application that uses LOB data, you can assess the compression by:

- Checking the size of the table space in Db2 once compression is used
- SMF record type 30. See the discussion of record type 30 in [“Assess compression of z/OS data” on page 13](#).

Chapter 7. Network transmission

Use IBM Sterling Direct:Connect for z/OS to transmit compressed data from z/OS to z/OS or z/OS to other platforms. Sterling Direct:Connect for z/OS is a separately orderable product.

For more information, see [KC for Sterling Direct:Connect \(www.ibm.com/docs/en/scfz/5.2.0\)](http://www.ibm.com/docs/en/scfz/5.2.0).

Request compression when transmitting data with SterlingDirect:Connect using the ZEDC initialization parameter.

Assessing compression: After transmitting data, you can assess the compression as follows.

- Synchronous execution:
 - SMF record type 113. See the discussion of SMF record type 113 in [“Assess compression of z/OS applications” on page 36](#).
- Asynchronous execution:
 - SMF record type 30. See the discussion of record type 30 in [“Assess compression of z/OS data” on page 13](#).
 - RMF EADM reporting (RMF 74.10). See [“RMF EADM reporting” on page 18](#).

Chapter 8. Programming

This section contains information about programming for Integrated Accelerator for zEDC.

MVS callable services

These are described later in this topic.

RMF

See [“Programming: RMF” on page 68](#).

zEnterprise Data Compression (zEDC)

Application interfaces for zEnterprise Data Compression

This topic describes the interfaces, considerations, and samples for zEnterprise Data Compression (zEDC).

Invoking unauthorized interfaces for zEnterprise Data Compression

zlib for zEnterprise Data Compression

The zlib data compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data. A modified version of the zlib compression library is used by zEDC. The IBM-provided zlib compatible C[®] library provides a set of wrapper functions that use zEDC compression when appropriate and when zEDC is not appropriate, software-based compression services are used.

The zlib wrapper functions use the following criteria to determine if zEDC can be used for compression:

- The system requirements for zEDC have been met.
- For a deflate stream, the parameters specified on `deflateInit2()` are supported by zEDC. For an inflate stream, all the parameters specified on `inflateInit2()` are supported.
- Because there are overhead costs when communicating with the hardware, on the first call to deflate or inflate a data stream, the provided input is checked to ensure that it is sufficiently large enough to make it worthwhile to use zEDC. If the data stream is large enough, zEDC is used. If the data stream is small, it might cost more to compress the data stream with zEDC so software-based compression services are used. This check is only performed on the first call to deflate or inflate a data stream.

If any of these criteria are not met, the zlib wrapper function calls the standard zlib functions to process the data stream in software.

Once zEDC is used as the compression mechanism (for example, after the first call to inflate or deflate the data stream is completed), you cannot change the compression method to software-based compression services. At the same time, if software-based compression services are used as the compression mechanism (for example, after the first call to inflate or deflate the data stream is completed), you cannot change the compression method to zEDC.

Note: Once a data stream starts using zEDC for compression, if a function is called that cannot be supported by zEDC or the zEDC hardware becomes unavailable, the unsupported function returns an error return code.

Standard zlib functions

The following table contains the standard zlib functions and whether they are supported using zEDC:

Table 5. Standard zlib functions and whether they are supported using zEDC		
zlib function	zEDC-supported	Details
zlibVersion	Supported.	Returns '1.2.11-zEDC'
deflateInit	Supported.	Level is ignored if using zEDC.
deflate	All flush modes are supported.	If the input buffer size is smaller than the minimum threshold for zEDC on the first call to deflate (compress) a data stream, the data stream is compressed using traditional software-based compression.
deflateEnd	Supported.	
inflateInit	Supported.	
inflate	Supported if the flush mode is one of the following: <ul style="list-style-type: none"> • z_no_flush • z_sync_flush • z_finish 	If either the input buffer size is smaller than a minimum threshold for zEDC or the flush mode is z_block or z_trees on the first call to inflate (decompress) a data stream, the data stream is decompressed using traditional software-based decompression. On subsequent calls to inflate a data stream, if the flush mode is z_block or z_trees and the stream is using zEDC decompression, Z_STREAM_ERROR is returned
inflateEnd	Supported.	
deflateInit2	Support is based on the input parameters.	<p>Input parameters:</p> <p>level This option is ignored for zEDC and does not affect the software or zEDC compression decision. This option is supported for zlib software compression.</p> <p>method Must be Z_DEFLATED.</p> <p>windowBits Must be -15 for raw deflate, 15 for zlib header and trailer, or 31 for gzip header and trailer. For all other windowBits values, the data stream uses traditional software-based compression.</p> <p>memLevel This option is ignored for zEDC and does not affect the software or zEDC compression decision. This option is supported for zlib software compression.</p> <p>strategy Use Z_DEFAULT_STRATEGY or Z_FIXED for zEDC. All other options use traditional software-based compression.</p>
deflateGetDictionary	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.
deflateSetDictionary	Supported.	This option is supported for zEDC when called before the first deflate call for the data stream and is not supported after the first call to deflate.
deflateCopy	Supported.	
deflateReset	Supported.	
deflateResetKeep	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.

Table 5. Standard zlib functions and whether they are supported using zEDC (continued)		
zlib function	zEDC-supported	Details
deflateParams	Support is based on the input parameters.	Input parameters: Level This option is ignored for zEDC. Strategy Use Z_DEFAULT_STRATEGY or Z_FIXED for zEDC. All other options use traditional software-based compression.
deflateTune	Supported.	This option only applies to traditional software-based compression. zEDC accepts the call, but none of the parameters apply to zEDC.
deflateBound	Supported.	
deflatePending	Supported.	
deflatePrime	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.
deflateSetHeader	Supported.	
inflateInit2	Supported.	
inflateGetDictionary	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.
inflateSetDictionary	Supported if called immediately after a call to inflate the data stream that returns Z_NEED_DICT.	Otherwise, Z_STREAM_ERROR is returned if the data stream is attempting to use zEDC decompression.
InflateSync	Supported.	
inflateSyncPoint	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.
inflateCodesUsed	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.
inflateCopy	Supported.	
inflateReset	Supported.	
inflatateReset2	Supported.	
inflatePrime	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC decompression.
inflateMark	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC decompression.
inflateGetHeader	Supported.	
inflateBackInit	Not supported for zEDC.	InflateBackInit forces stream to software-based compression.
inflateBack	Not supported for zEDC.	
inflateValidate	Not supported for zEDC.	Returns Z_STREAM_ERROR if the stream is using zEDC.
zlibCompileFlags	Supported.	

<i>Table 5. Standard zlib functions and whether they are supported using zEDC (continued)</i>		
zlib function	zEDC-supported	Details
compress	Supported.	
compress2	Supported.	Level is ignored if using zEDC.
compressBound	Supported.	
uncompress	Supported.	
uncompress2	Supported.	
gz* routines	Not supported for zEDC.	Uses software-based compression for inflate and deflate functions.
checksum functions	Not supported for zEDC.	Checksum functions calculate the checksum values using software-based compression services.

IBM-provided zlib compatible C library

The IBM-provided zlib compatible C library provides the following query functions in addition to the standard zlib functions:

deflateHwAvail(*buflen*)

Determines if the compression accelerator is available for a deflate operation. The input parameter *buflen* is an integer that represents the input buffer size of the first deflate request. The function returns an integer with a value of 1 if the compression accelerator will be used for the deflate operation or a value of 0 if software will be used instead.

inflateHwAvail(*buflen*)

Determines if the compression accelerator is available for an inflate operation. The input parameter *buflen* is an integer that represents the input buffer size of the first inflate request. The function returns an integer with a value of 1 if the compression accelerator will be used for this inflate operation or a value of 0 if software will be used instead.

hwCheck(*strm*)

Determines if a zlib stream is using the compression accelerator or software compression. The input parameter *strm* is a pointer to a zlib *z_stream* structure to check. The function returns an integer with a value of 0 if the stream has gone to the compression accelerator, a value of 1 if the stream is pending to go to the compression accelerator, but still could fall back to software compression, a value of 2 if the stream has gone to software compression, or *Z_STREAM_ERROR* if the stream has not been initialized correctly.

Running zlib

To compress data with zEDC, your installation must meet the system requirements.

To use the IBM-provided zlib compatible C library for data compression or data expansion services, follow these steps:

1. Link or relink applications to use the IBM-provided zlib.

The IBM-provided zlib is an archive file in the z/OS UNIX System Services file system and can be statically or dynamically linked into your applications. The paths for the zlib archive file and the zlib header files are:

Path for the zlib archive file:

/lib/libz.a

Path for 31-bit non-xplink dynamic library files:

/lib/libz.so

/lib/libz.x

Path for 31-bit xplink dynamic library files:

/lib/libzX.so

/lib/libzX.x

Path for 64-bit dynamic library files:

/lib/libz64.so

/lib/libz64.x

Path for the zlib header files:

/usr/include/

When a new IBM service is provided for zlib, all applications that statically or dynamically link zlib must relink in order to use the updated IBM-provided zlib and take advantage of the new function.

2. Provide access to System Authorization Facility (SAF):

- Access to zEDC Express is protected by the SAF FACILITY resource class for installations that are running IBM zEnterprise z14 and earlier processors. IBM zEnterprise z15 and later processors no longer have this requirement.
- Give READ access to FPZ.ACCELERATOR.COMPRESSION to the identity of the address space that the zlib task will run in.

The access check is performed during the first call in a given task. The results of that first check are cached for the duration of the task.

3. Use the z/OS UNIX environmental variable `_HZC_COMPRESSION_METHOD` to control if zEDC is used for data compression. If the value of *software* is set, software-based compression services are used. All other values result in the default behavior of attempting to use zEDC for data compression.
4. Ensure that adequately sized input buffers are available. If the input buffer size falls below the minimum threshold, data compression occurs using zlib software compression and not zEDC.

Note: IBM zEnterprise z15 and above processor thresholds will no longer be tunable through parmlib. The IQPPRMxx will still be allowed in the configuration, but the values are not accepted.

The environment variables `_HZC_DEFLATE_THRESHOLD` and `_HZC_INFLATE_THRESHOLD` can also be used to control the threshold for going to zEDC. The valid values are in the range 1-9999999.

For example:

`_HZC_DEFLATE_THRESHOLD=1` would force all deflate requests with an initial input size of 1 byte or larger to use zEDC.

This threshold can be controlled at a system level by using the PARMLIB member IQPPRMxx.

5. Use the z/OS UNIX environmental variable, `_HZC_CHECKSUM_METHOD`, to control if SIMD acceleration is used in checksum verification. If the value of *software* is set, software-based checksum verification is used. All other values result in the default behavior, which means if hardware supported SIMD, then SIMD acceleration is used.
6. Allocate the correct amount of storage for I/O buffers. The zEDC requests generated by zlib use predefined I/O buffer pools. The size of these I/O buffer pools can be set using PARMLIB member IQPPRMxx.

For IBM z15 and later processors, these buffers are no longer applicable. The IQPPRMxx will still be allowed in the configuration, but the values will no longer be accepted.

7. Environmental variable operations are not thread-safe. For multi-thread programming, if some threads use zEDC while other threads do not, a thread-specific data key (`hzc_compression_method`) is provided to handle this scenario.
- a. To use the new feature, the user code must include the following statement: `extern pthread_key_t hzc_compression_method;`
 - b. The key must be created in the user code and data must be specified to the key. The destructor routine is also defined by the user.
 - c. If any data is specified to the thread-specific data key `hzc_compression_method`, its value will override the environmental variable `_HZC_COMPRESSION_METHOD`. Otherwise, `_HZC_COMPRESSION_METHOD` will take effect.

- d. If the specified data value is software, software-based data compression is used. All other values result in the default behavior of attempting to use zEDC for data compression.

For more information about thread-specific data keys, see [pthread_key_create\(\) — Create thread-specific data key](#) in *z/OS C/C++ Runtime Library Reference*.

When zlib is statically linked into an application that runs on software or hardware that is not compatible with zEDC, zlib uses the following compression and decompression:

Table 6. Compression and decompression with zlib			
Hardware level	z/OS level	zEDC Express	Description
zEC12 (with GA2 level microcode)	z/OS V2R1	Active	zEDC is used for both data compression and decompression.
zEC12 (with GA2 level microcode)	z/OS V2R1	Not Active	Requirements are not met for zEDC. When zEDC Express is not available, traditional software zlib is used for compression and decompression.
Pre-zEC12 (with GA2 level microcode)	z/OS V2R1 or pre-z/OS V2R1	N/A	Requirements are not met for zEDC. When zEDC Express is not available, traditional software zlib is used for compression and decompression.

zEDC error handling:

- If a z System compression accelerator is unavailable, data compression requests transfer to another z System compression accelerator configured to the same partition. These request transfers are transparent to the application.
- If all z System compression accelerators are unavailable, an error message is sent to the application.

Invoking z System authorized interfaces for zEnterprise Data Compression

To compress data with zEDC, your installation must meet the system requirements.

All z/OS exploitation of zEDC handles mixed hardware and software levels. Compatibility APAR OA41245 provides software decompression for installations running with z/OS V1R13 or V1R12. The same software decompression is also provided for installations running z/OS V2R1 on pre-IBM zEnterprise EC12 (with GA2 level microcode). This allows access to compressed data on all combinations of environments.

Table 7. Compression and decompression with z System authorized interfaces for zEDC			
Hardware level	z/OS level	zEDC Express	Description
zEC12 (with GA2 level microcode)	z/OS V2R1	Active	zEDC is used for both data compression and decompression.
zEC12 (with GA2 level microcode)	z/OS V2R1	Not Active	Requirements are not met for zEDC. Software-based decompression services for zEDC Express compressed data are used because zEDC Express compression is not available.
Pre-zEC12 (with GA2 level microcode)	z/OS V2R1	N/A	Requirements are not met for zEDC. Software-based decompression services for zEDC Express compressed data are used because zEDC Express compression is not available.

Table 7. Compression and decompression with z System authorized interfaces for zEDC (continued)			
Hardware level	z/OS level	zEDC Express	Description
Pre-zEC12 (with GA2 level microcode)	Pre-z/OS V2R1	N/A	Requirements are not met for zEDC. Software-based decompression services for zEDC Express compressed data are used because zEDC Express compression is not available. APAR OA41245 is required to use the software-based decompression services.

z System authorized compression services

This topic describes the compression services that are available when using z System authorized interfaces for zEDC.

FPZ4RZV - Rendezvous compression service

Description

The FPZ4RZV service performs the required setup and initialization of the compression services for an exploiter. The scope is the address space of the application and it is valid for the life of the Cross Memory Resource Owner (CMRO) task.

Notes:

1. A maximum of 255 rendezvous tokens are supported per each address space. This allows multiple applications to exploit the compression driver so each can maintain their own rendezvous scope.
2. All 64-bit storage is obtained with the MEMLIMIT=NO option.

Table 8. Environment for the FPZ4RZV service	
Environmental factor	Requirement
Minimum authorization:	Supervisor State with Key 0
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	64-bit
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Table 9. Parameters for the FPZ4RZV service			
Name	Type	Input/Output	Description
<i>ApplicationId</i>	Fixed(32)	Input	The application type to use. 0x01 is the application type for zEDC.

Table 9. Parameters for the FPZ4RZV service (continued)

Name	Type	Input/ Output	Description
FPZ4RZV_options	Bit(64)	Input	Options for the FPZ4RZV service: SoftwareInflate (X'80000000 00000000') Allows compression requests to fall back to software inflation when no compression devices are available. EnableABCScatter (X'40000000 00000000') Allows compression requests to use the FPZ4ABC compression service to submit work with scatter/gather lists. FailOnNoDevices (X'20000000 00000000') If specified, compression requests fail when no compression devices are available. If FailOnNoDevices is not specified, a valid rendezvous token is returned even if no compression devices are currently available. This returned rendezvous token is used for all other services. PlusOne (X'08000000 00000000') If specified, compression requests will only use zEDC Express Adapters with the February 26, 2014 Firmware MCL release, or later. RmrEntriesExact (X'04000000 00000000') If specified, the <i>rmr_entries</i> parameter represents the maximum number of outstanding memory registrations for this rendezvous. If this limit is exceeded, the FPZ4RMR service may fail the request.
<i>userid</i>	Char(8)	Input	An eight character EBCDIC string identifying the user.
<i>rmr_entries</i>	Fixed(32)	Input	The estimated number of FPZ4RMR compression service calls to be performed that helps to size the tables used until the maximum number of registrations is reached. This is an optional parameter. The value of the <i>rmr_entries</i> parameter can be anywhere between 1 and 64K. The default is 128. Define <i>rmr_entries</i> as integer data of length 32.
Rendezvous token	Char(16)	Output	This is the token that must be passed to all FPZ services.
Return code	Fixed(31)	Output	The return code for the service.
Reason code	Fixed(32)	Output	The reason code for the service.

Table 10. Return and reason codes for the FPZ4RZV service		
Hexadecimal return code	Reason code	Meaning and action
00	0000	Meaning: The call completed successfully. Action: None.
04	0000	Meaning: No zEDC devices are available. zEDC support is active so it is possible that zEDC devices might become available in the future. Action: If zEDC devices are available to this system, perform diagnostics to determine the reason for the failure.
04	0102	Meaning: No zEDC devices are available because the system requirements for zEDC were not met. A 'thin' rendezvous was created. Action: None.
08	0000	Meaning: No zEDC devices are available because the system requirements for zEDC were not met. This is the result of RvzFailOnNoDev being ON or SoftwareInflate being OFF when on downlevel hardware or software. No rendezvous token is returned. Action: None.
0C	0201	Meaning: Invalid parameter combination. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0207	Meaning: The calling environment is invalid. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0210	Meaning: <i>rmr_entries</i> specified an invalid value. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0226	Meaning: Invalid application specified. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
10	0301	Meaning: An internal error caused recovery to be entered. Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.
10	0303	Meaning: The maximum number of rendezvous tokens have been reached for the address space. Action: Determine if the calling program is at fault because of a coding error. If there is no coding error, another program might be consuming all the rendezvous tokens for the address space. Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

Table 10. Return and reason codes for the FPZ4RZV service (continued)		
Hexadecimal return code	Reason code	Meaning and action
10	030B	Meaning: The CMRO task is ending. Action: None, since the address space is ending. This reason code should be accounted for in your code scenarios.
10	030C	Meaning: Too many latch sets requested. Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.

FPZ4PRB — Probe device availability compression service

Description

The FPZ4PRB service checks for the required hardware and software needed for zEDC. This service returns successful if they are available to the system.

Table 11. Environment for the FPZ4PRB service	
Environmental factor	Requirement
Minimum authorization:	Supervisor State with Key 0
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN=SASN
AMODE:	64-bit
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Table 12. Parameters for the FPZ4PRB service			
Name	Type	Input/Output	Description
<i>ApplicationId</i>	Fixed(3 2)	Input	The application type to use. 0x01 is the application type for zEDC.
FPZ4PRB_options	Bit(64)	Input	Options for the FPZ4PRB service: PlusOne (X'80000000 00000000') If specified, only zEDC Express Adapters with the March 31, 2014 Firmware MCL release, or later, will be honored. The value returned in <i>NumDevices</i> will only indicate this subset of devices. Note: This option is not applicable when running on z15 and above. PRBHasSync(X'40000000) If specified, the service will return with a return code of zero in the event that zEDC is running on a z15.
<i>NumDevices</i>	Fixed(3 2)	Output	The number of devices available for this application.

Table 12. Parameters for the FPZ4PRB service (continued)			
Name	Type	Input/Output	Description
Return code	Fixed(3 1)	Output	The return code for the service.
Reason code	Fixed(3 2)	Output	The reason code for the service.

Table 13. Return and Reason Codes for the FPZ4PRB service		
Hexadecimal Return Code	Reason Code	Meaning and Action
00	0000	Meaning: Devices are available. Action: None.
00	00	Meaning: Compression is available. Action: None.
08	0900	Meaning: The z/OS software level is not correct for zEDC. Action: None.
08	0901	Meaning: The hardware level is not correct for zEDC. Action: None.
08	0902	Meaning: No zEDC devices are available. The hardware is at the correct level, but no zEDC devices were available. Action: If zEDC devices are available to this system, perform diagnostics to determine the reason for the failure.
08	0903	Meaning: zEDC devices were available during this IPL at some point, but there are no zEDC devices available now. Action: Perform diagnostics to determine the reason for the failure.

FPZ4RMR - Memory registration compression service

Description

The FPZ4RMR service registers a segment of memory for use by zEDC. The result is that this storage becomes fixed. The data area passed to FPZ4RMR must be page-aligned, and the size must be a multiple of a page boundary.

Note: This is not compatible with existing page fix services. This storage is eligible to be used for I/O as a result of this service.

Table 14. Environment for the FPZ4RMR service	
Environmental factor	Requirement
Minimum authorization:	Supervisor State with Key 0
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN=SASN
AMODE:	64-bit

Table 14. Environment for the FPZ4RMR service (continued)

Environmental factor	Requirement
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Table 15. Parameters for the FPZ4RMR service

Name	Type	Input/ Output	Description
<i>ApplicationId</i>	Fixed(32)	Input	The application type to use. 0x01 is the application type for zEDC.
FPZ4RMR_options	Bit(64)	Input	There are no supported options for the FPZ4RMR service.
Rendezvous token	Char(16)	Input	The rendezvous token.
Data@	Ptr(64)	Input	The address of the data area to register. Note: Large page frames must be in fixed storage.
DataLen	Fixed(64)	Input	The length of the data area to register.
Reserved	Fixed(32)	Input	Reserved. Must be 0.
DataKey	Fixed(8)	Input	The key of the data area to register. The format of this parameter is 0xk0, where <i>k</i> represents the key of the data area.
RMR Token	Char(8)	Output	The region memory registration token associated with this data area. This token needs to be passed to the FPZ4ABC service when this data area is used as input or output.
Return code	Fixed(31)	Output	The return code for the service.
Reason code	Fixed(32)	Output	The reason code for the service.

Table 16. Return and Reason Codes for the FPZ4RMR service

Hexadecimal Return Code	Reason Code	Meaning and Action
00	0000	Meaning: The call completed successfully. Action: None.
08	0000	Meaning: Memory can not be registered because of lack of hardware support. Action: None.
08	0900	Meaning: Incorrect software level for zEnterprise data compression accelerator support. Action: None.

Table 16. Return and Reason Codes for the FPZ4RMR service (continued)		
Hexadecimal Return Code	Reason Code	Meaning and Action
0C	0207	Meaning: The calling environment is invalid. Action: Determine if the calling program is at fault because of a coding error.
0C	0208	Meaning: An invalid rendezvous token was passed. Action: Check that the application successfully called the FPZ4RZV service.
0C	021D	Meaning: The supplied region was not CONTROL(AUTH). Action: Determine if the calling program is at fault because of a coding error.
0C	021E	Meaning: The supplied region address is incorrect. It might not have been page-aligned. Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.
0C	021F	Meaning: The region length is invalid. It is possible that it is not a multiple of page size. Action: Determine if the calling program is at fault because of a coding error.
0C	0220	Meaning: There is a region key mismatch. Action: Determine if the calling program is at fault because of a coding error.
0C	0226	Meaning: An invalid application ID was encountered. Action: Determine if the calling program is at fault because of a coding error.
0C	0227	Meaning: Rendezvous was not created with data space support. Action: Determine if the calling program is at fault because of a coding error.
10	0301	Meaning: An internal error has occurred. Action: Determine if the calling program is at fault because of a coding error.
10	0304	Meaning: Compression services were not initialized. Rendezvous was not called. Action: Check that the application successfully called the FPZ4RZV service.
10	0305	Meaning: Capacity has been reached for memory registrations. Action: Determine if the calling program is at fault because of a coding error.

Table 16. Return and Reason Codes for the FPZ4RMR service (continued)

Hexadecimal Return Code	Reason Code	Meaning and Action
10	0306	Meaning: There is not enough DMA memory available. Action: Determine if the calling program is at fault because of a coding error.
10	030D	Meaning: Missing latch set token. Action: Determine if the calling program is at fault because of a coding error.

*FPZ4DMR - Deregister memory compression service**Description*

The FPZ4DMR service unregisters a segment of memory for use by zEDC Express. The result is that this storage becomes unfixed.

Table 17. Environment for the FPZ4DMR service

Environmental factor	Requirement
Minimum authorization:	Supervisor State
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN=SASN
AMODE:	64-bit
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Table 18. Parameters for the FPZ4DMR service

Name	Type	Input/Output	Description
<i>ApplicationId</i>	Fixed(3 2)	Input	The application type to use. 0x01 is the application type for zEDC.
FPZ4DMR_options	Bit(64)	Input	There are no supported options for the FPZ4DMR service.
Rendezvous token	Char(1 6)	Input	The rendezvous token.
RMR token	Char(8)	Input	The region memory registration (RMR) token associated with this data area to be unregistered.
Return code	Fixed(3 1)	Output	The return code for the service.
Reason code	Fixed(3 2)	Output	The reason code for the service.

Table 19. Return and Reason Codes for the FPZ4DMR service		
Hexadecimal Return Code	Reason Code	Meaning and Action
00	0000	Meaning: The call completed successfully. Action: None.
08	0900	Meaning: Incorrect software level for zEnterprise data compression accelerator support. Action: None.
0C	0207	Meaning: The calling environment is invalid. Action: Determine if the calling program is at fault because of a coding error.
0C	0208	Meaning: An invalid rendezvous token was passed. Action: Check that the application successfully called the FPZ4RZV service.
0C	0209	Meaning: An invalid RMR token was provided. Action: Determine if the calling program is at fault because of a coding error.
10	0301	Meaning: An internal error has caused recovery to be entered. Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.
10	0304	Meaning: Compression services were not initialized. Rendezvous was not called. Action: Check that the application successfully called the FPZ4RZV service.
10	030D	Meaning: Missing latch set token. Action: Determine if the calling program is at fault because of a coding error.

FPZ4ABC – Submit compression request

Description

The FPZ4ABC service submits a single autonomous compression request for one or more DEFLATE blocks. The input and output buffers can be either direct buffers or scatter/gather lists. The maximum size of a request for FPZ4ABC is 1 MB.

Table 20. Environment for the FPZ4ABC service	
Environmental factor	Requirement
Minimum authorization:	Supervisor State with Key 0
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN=SASN
AMODE:	64-bit
Interrupt status:	Enabled for I/O and external interrupts

Table 20. Environment for the FPZ4ABC service (continued)

Environmental factor	Requirement
Locks:	No locks held

Table 21. Parameters for the FPZ4ABC service

Name	Type	Input/Output	Description
<i>ApplicationId</i>	Fixed(32)	Input	The application type to use. 0x01 is the application type for zEDC.
FPZ4ABC_options	Bit(64)	Input	Options for the FPZ4ABC service: Inflate (X'80000000 00000000') When ON, specifies that this is an inflation request. Input Scatter List (X'40000000 00000000') When ON, the area pointed to by input@ is a scatter/gather list. Output Scatter List (X'20000000 00000000') When ON, the area pointed to by output@ is a scatter/gather list. AbcSyncRequest(X'10000000 ..) When ON, this is the synchronous request.
Rendezvous token	Char(16)	Input	The rendezvous token.
Input@	Ptr(64)	Input	The address of the input area or input scatter/gather list.
Output@	Ptr(64)	Input	The address of the output area or output scatter/gather list.
Input@RMR Token	Char(8)	Input	The region memory registration (RMR) token for the input area or area pointed to by the input scatter/gather list.
Output@RMR Token	Char(8)	Input	The region memory registration (RMR) token for the output area or area pointed to by the output scatter/gather list.
InputLen	Fixed(64)	Input	The length of the area pointed to by Input@. In the event that a scatter/gather list was provided using Input@, the total length of the areas provided by the scatter/gather areas must be provided.
OutputLen	Fixed(64)	Input	The length of the area pointed to by Output@. In the event that a scatter/gather list was provided using Output@, the total length of the areas provided by the scatter/gather areas must be provided.
GeneratedOutputLen	Fixed(64)	Output	This length describes how much output was generated and stored in either the Output@ or the scatter/gather list specified by Output@. This length spans across scatter/gather entries.
Return code	Fixed(31)	Output	The return code for the service.

Table 21. Parameters for the FPZ4ABC service (continued)

Name	Type	Input/Output	Description
Reason code	Fixed(32)	Output	The reason code for the service.

The FPZ4ABC service allows for the input and output areas to span several non-contiguous areas. The header of the FPZ4ABC list is immediately followed by the list entries. All entries in the scatter/gather list must be associated with the same RMR token.

Scatter/gather lists have alignment rules and every entry in the scatter/gather list is checked for the following conditions:

- The start of the first buffer in the list can be on any byte boundary.
- The end of the first buffer must be on the required byte boundary.
- The start / end of the intermediate buffers must be on the required byte boundary.
- The start of the last buffer must be on the required byte boundary.
- The end of the last buffer can be on any boundary.

All required boundaries are on 128-byte alignment. A maximum of 8 scatter/gather entries are allowed.

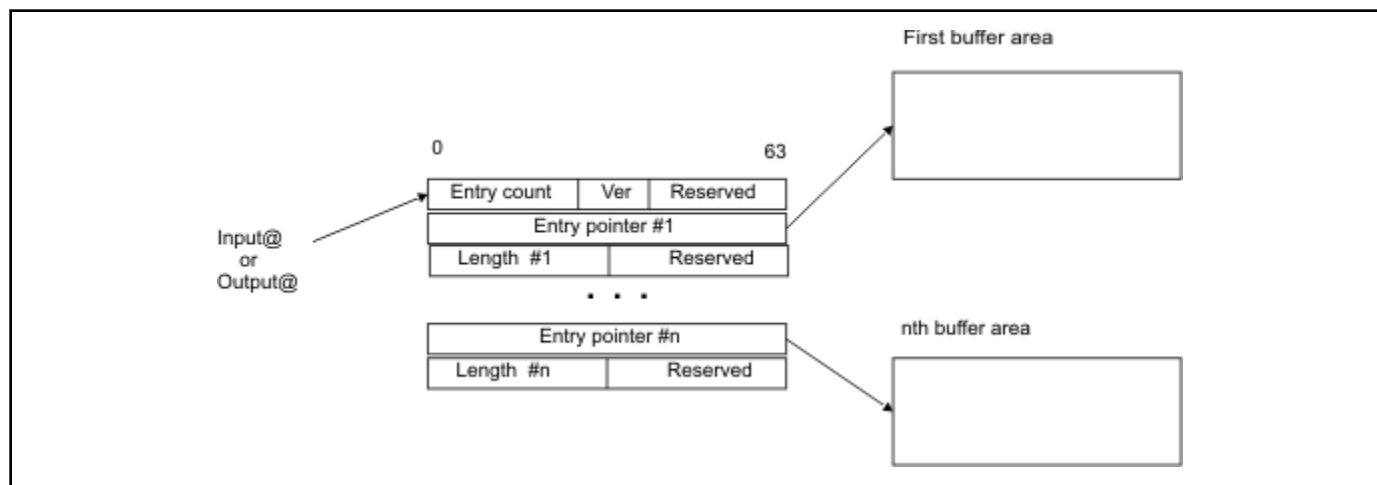


Table 22. Header elements in the FPZ4ABC-generated list

Name	Type	Description
# Of Entries	Fixed(32)	The number of entries in the list.
Version	Fixed(8)	The version associated with the list.
Reserved	Char(3)	Reserved space.

Table 23. Entries elements in the FPZ4ABC-generated list

Name	Type	Description
Address	Fixed(64)	The address into the area mapped by the region memory registration (RMR) token.
Length	Fixed(32)	The length of the area, starting at address, to use.
Reserved	Fixed(32)	Reserved space.

Table 24. Return and Reason Codes for the FPZ4ABC service

Hexadecimal Return Code	Reason Code	Meaning and Action
00	0000	Meaning: The call completed successfully. Action: None.
04	2000	Meaning: No zEDC devices are available. Inflate is completed in software when hardware is not available. Action: None.
08	0000	Meaning: No zEDC devices are available. Action: If zEDC devices are available to this system, perform diagnostics to determine the reason for the failure.
0C	0202	Meaning: One of the buffers had a length of 0, or the first word of a length was non-zero, or one of the buffers has a length greater than 1 MB. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0203	Meaning: A failure occurred while accessing one of the provided scatter/gather buffers. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0206	Meaning: The output area was not large enough to complete the request. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0207	Meaning: The calling environment is invalid. The caller is either Problem State, non-zero key, or in XMEM mode. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0208	Meaning: The rendezvous token is invalid. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0209	Meaning: The region memory registration (RMR) token is invalid. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0221	Meaning: The header of the FPZ4ABC-generated list was not formed correctly. Action: Check the calling program for a probable coding error. Correct the program and rerun it.

Table 24. Return and Reason Codes for the FPZ4ABC service (continued)		
Hexadecimal Return Code	Reason Code	Meaning and Action
0C	0222	<p>Meaning: Either zero or a number greater than the maximum supported was specified for the number of entries in the FPZ4ABC-generated list.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	0223	<p>Meaning: A buffer in the scatter/gather list was not aligned properly.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	0224	<p>Meaning: The total length of the buffers in the scatter/gather list does not match the length in the parmlist.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	0225	<p>Meaning: Scatter/gather was requested, but it was not enabled for this rendezvous token.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	1202	<p>Meaning: An address range is not contained in the region denoted by the region memory registration (RMR) token.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	1203	<p>Meaning: An unsupported operation was requested.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	1205	<p>Meaning: An inflate request failed because of malformed data.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	2101	<p>Meaning: An inflate request failed in software mode due to malformed input data.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
0C	2102	<p>Meaning: Not enough space in the output buffer to process the request in software mode.</p> <p>Action: Check the calling program for a probable coding error. Correct the program and rerun it.</p>
10	0301	<p>Meaning: An internal component error occurred.</p> <p>Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.</p>

Table 24. Return and Reason Codes for the FPZ4ABC service (continued)		
Hexadecimal Return Code	Reason Code	Meaning and Action
10	0304	Meaning: A rendezvous has not yet occurred for this address space. Action: Check that the application successfully called the FPZ4RZV service.
10	1203	Meaning: There are no zEDC devices available and either the request was a deflate request or software inflate was not enabled. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
10	1301	Meaning: The request failed unexpectedly for an unknown reason. Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.
10	030D	Meaning: Missing latch set token. Action: Determine if the calling program is at fault because of a coding error.

FPZ4URZ - Unrendezvous compression request

Description

The FPZ4URZ service removes the address space level information related to zEDC Express compression services. Any outstanding memory registrations are unregistered.

Table 25. Environment for the FPZ4URZ service	
Environmental factor	Requirement
Minimum authorization:	Supervisor State
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	64-bit
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held

Table 26. Parameters for the FPZ4URZ service			
Name	Type	Input/Output	Description
<i>ApplicationId</i>	Fixed(32)	Input	The application type to use. 0x01 is the application type for zEDC.
FPZ4URZ_options	Bit(64)	Input	There are no supported options for the FPZ4URZ service.
Rendezvous token	Char(16)	Input	The rendezvous token.

Table 26. Parameters for the FPZ4URZ service (continued)

Name	Type	Input/Output	Description
Return code	Fixed(3 1)	Output	The return code for the service.
Reason code	Fixed(3 2)	Output	The reason code for the service.

Table 27. Return and Reason Codes for the FPZ4URZ service

Hexadecimal Return Code	Reason Code	Meaning and Action
00	0000	Meaning: The call completed successfully. Action: None.
0C	0207	Meaning: The calling environment is invalid. The caller is either Problem State, non-zero key, or in XMEM mode. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
0C	0208	Meaning: An invalid rendezvous token was passed. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
10	0301	Meaning: An internal error has caused recovery to be entered. Action: Search problem reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center.
10	0304	Meaning: Compression services were not initialized. Action: Check the calling program for a probable coding error. Correct the program and rerun it.
10	030D	Meaning: Missing latch set token. Action: Determine if the calling program is at fault because of a coding error.

Usage example of a z System authorized service

The following example uses the authorized services to perform compression using zEDC Express. If zEDC Express adapters are not available, data is written to the destination uncompressed.

The FPZ4PRB service is called intermittently after the FPZ4ABC service returns to the application with a return code that indicates that all zEDC devices have left the configuration.

```

Call FPZ4RZV(AppId, RzvOptions, RzvUserId, RzvToken, RetCode, RsnCode)          /* Rendezvous
with the compression                                                         device
driver (once per address                                                         space) */

If RetCode = RcNoDevices Then                                                  /* If no
devices available */                                                         /* Indicate no
  NoDevices = ON
devices */

Call FPZ4RMR(AppId, RmrOptions, RzvToken, InBuffer@, InBufferLen, 0, InBufKey, InRmrToken,
RetCode, RsnCode)                                                            /* Register the
input buffer */
Call FPZ4RMR(AppId, RmrOptions, RzvToken, OutBuffer@, OutBufferLen, 0, OutBufKey, OutRmrToken,

```

```

        RetCode, RsnCode)                                /* Register the
output buffer for                                         compressed
data */
Do Until End of Data
    Read next block of data into InBuffer@
        If NoDevices = ON Then                            /* If no
devices available */                                     /* If no
        Call FPZ4PRB(AppId, Options, NumDevices, RetCode, RsnCode) /* Probe for
new devices */
        If RetCode = RcOk Then                            /* If devices
now available */                                         /* If devices
        NoDevices = OFF                                    /* Indicate we
have devices */                                         /* Indicate we
        Else                                              /* Else no
devices */                                              /* Else no
        Write InBuffer                                    /* Processed
uncompressed data */
        If NoDevices = OFF Then                            /* If devices
available */                                             /* If devices
        Call FPZ4ABC(RzvToken,                            /* Perform
            InBuffer@, InBufferLen, InRmrToken,
            OutBuffer@, OutBufferLen, OutRmrToken,
            RetCode, RsnCode)
compression */
        If RetCode = RcOk Then                            /* If data was
compressed */                                         /* If data was
        Write OutBuffer                                    /* Process
compressed data */                                     /* Process
        Else If RetCode = RcNoDevices Then                /* If no
devices available */                                     /* If no
        NoDevices = ON                                    /* Indicate no
devices */                                              /* Indicate no
        Write InBuffer                                    /* Process
uncompressed data */
End Loop
Call FPZ4DMR(AppId, DmrOptions, RzvToken, InRmrToken, RetCode, RsnCode)
Call FPZ4DMR(AppId, DmrOptions, RzvToken, OutRmrToken, RetCode, RsnCode)

```

Troubleshooting for zEDC

This topic explains troubleshooting techniques for zEDC.

RMF provides the following data for the z System accelerator device:

- Load current partition is putting on device
- Compression and decompression request rate and throughput
- Achieved compression ratio

See *z/OS Resource Measurement Facility User's Guide* for the available options to specify on your Monitor I session for reporting on the z System compression accelerator.

Programming: RMF

This section contains information about RMF considerations for programming for Integrated Accelerator for zEDC, including ERBSCMG3 and SCM EADM.

ERBSCMG3 - Extended Asynchronous Data Mover (EADM) data table

Dec offset	Hex offset	Name	Length	Format	Description
SCMG3 header section:					
0	0	SCM_EyeC	5	EBCDIC	Acronym SCMG3
5	5	SCM_Ver	1	binary	SCMG3 version

Dec offset	Hex offset	Name	Length	Format	Description
6	6	SCM_Flag	1	binary	SCMG3 flags Bit Meaning when set 0 SCMG3 converted to lower service level 1 SCMG3 converted to higher release or service level 2–7 Reserved
7	7	SCM_VerGat	1	binary	Original version of SCMG3 before data conversion
8	8	SCM_HdrL	4	binary	Length of SCMG3 header
12	C	SCM_TotL	4	binary	Total length of SCMG3
16	10	*	2	*	Reserved
18	12	SCM_DIL	2	binary	Length of SCM DI entry
20	14	SCM_DIO	4	binary	Offset to SCM DI entry
24	18	SCM_CML	2	binary	Length of SCM CM entry
26	1A	SCM_CMN	2	binary	Number of SCM CM entries
28	1C	SCM_CMO	4	binary	Offset to SCM CM entries
EADM device information entry (SCM DI):					
0	0	SCMDI_EyeC	5	EBCDIC	Eyecatcher SCMDI
5	5	SCMDI_Ver	1	binary	SCMDI version
6	6	SCMDI_VerGat	1	binary	Original version of SCMDI before data conversion
7	7	*	1	*	Reserved
8	8	SCMDI_Body	80	binary	EADM device information data For information about all fields contained in the SCMDI_Body section, see SMF record type 74 subtype 10, Extended asynchronous data mover (EADM) device (subchannel) information in z/OS MVS System Management Facilities (SMF) . Note that SCMDI_Body is at offset 8 in the EADM Device Information Entry section of SCMG3. This offset must be added to offsets of R7410DI when accessing fields of SCMDI in SCMG3.
8	8	R7410DSCT	4	binary	SSCH count across all devices
12	C	R7410DNUM	4	binary	Number of updates to the time accumulation fields
16	10	R7410DFPT	8	binary	Sum of function pending times across all devices in units of 128 microseconds (doubleword format).
24	18	R7410DIQT	8	binary	Sum of IOP queue times across all devices in units of 128 microseconds (doubleword format).
32	20	R7410DCRT	8	binary	Sum of initial command response times across all devices in units of 128 microseconds (doubleword format).
SCM configuration measurement entry (SCM CM):					
0	0	SCMCM_EyeC	5	EBCDIC	Eyecatcher SCMCM
5	5	SCMCM_Ver	1	binary	SCMCM version
6	6	SCMCM_VerGat	1	binary	Original version of SCMCM before data conversion
7	7	*	1	*	Reserved

Dec offset	Hex offset	Name	Length	Format	Description
8	8	SCMCM_Body	56	binary	SCM configuration measurement data For information about all fields contained in the SCMCM_Body section, see SMF record type 74 subtype 10, Storage class memory (SCM) configuration measurement section in z/OS MVS System Management Facilities (SMF) . Note that SCMCM_Body is at offset 8 in the SCM Configuration Measurement Entry section of SCMG3. This offset must be added to offsets of R7410CM when accessing fields of SCMCM in SCMG3.
8	8	R7410CRID	2	binary	SCM resource identifier
10	A	R7410CPID	2	binary	Part identifier
12	C	R7410CDUS	4	binary	Data unit size in bytes
16	10	R7410CRQC	4	binary	Internal requests processed at CPC level
20	14	R7410CRQ	4	binary	Internal requests processed at LPAR level
24	18	R7410CDWC	4	binary	Data units written at CPC level
28	1C	R7410CDW	4	binary	Data units written at LPAR level
32	20	R7410CDRC	4	binary	Data units read at CPC level
36	24	R7410CDR	4	binary	Data units read at LPAR level
40	28	R7410CRTC	4	binary	Aggregate time spent on execution of requests involving resource part in units of 128 microseconds at CPC level
44	2C	R7410CRT	4	binary	Aggregate time spent on execution of requests involving resource part in units of 128 microseconds at LPAR level
48	30	R7410CIQC	4	binary	Accumulated IOP queue time in units of 128 microseconds at CPC level
52	34	R7410CWUC	4	binary	Utilization at CPC level.
56	38	R7410CWU	4	binary	Utilization at LPAR level.
60	3C	R7410FLG	1	binary	Flag byte: Bit Meaning when set 0 SCM resource type is VFM 1–7 Reserved
61	3D	*	3	*	Reserved

EADM - Tabular report data table ERBSCMT3

RMF builds ERBSCMT3 when using EADM as a report type.

Name	Type	Meaning	Report
SCMDTLN	K	Logical line number	-
SCMDTPSN	K	Sequence number	-
SCMRPID	N	Card id	Yes
SCMUTL	N	LPAR utilization percentage	Yes
SCMUTLC	N	Total utilization percentage	Yes
SCMDRD	N	LPAR bytes read per second	Yes
SCMDRDC	N	Total bytes read per second	Yes

Name	Type	Meaning	Report
SCMDWR	N	LPAR bytes written per second	Yes
SCMDWRC	N	Total bytes written per second	Yes
SCMQR	N	LPAR requests processed per second	Yes
SCMQRC	N	Total requests processed per second	Yes
SCMART	N	LPAR response time per request in milliseconds	Yes
SCMARTC	N	Total response time per request in milliseconds	Yes
SCMAQTC	N	Total IOP queue time per request in milliseconds	Yes
SCMTRQ	N	LPAR number of requests	Util
SCMTRQC	N	Total number of requests	Util
SCMHSCR	N	Number of SSCH instructions to all EADM devices per second	Yes
SCMHSCH	N	Total number of SSCH instructions to all EADM devices	Yes
SCMHFPT	N	Function pending time across all EADM devices in milliseconds	Yes
SCMHIQT	N	IOP queue time across all EADM devices in milliseconds	Yes
SCMHCRT	N	Command response time across all EADM devices in milliseconds	Yes

Chapter 9. Messages

This topic includes the messages that are new or changed for Integrated Accelerator for zEDC.

RMF (ERB and GPM) messages

This topic describes the RMF messages that are affected by Integrated Accelerator for zEDC.

ERB629I **EADM activity data is currently not available.**

Explanation

Within the current report interval, RMF can not report any extended asynchronous data mover (EADM) activity. This could be caused by any of the following reasons:

- There was no EADM activity.
- EADM activity data was not collected (Monitor III Gatherer option NOEADM was specified).
- EADM is not supported by the processor.
- There is backlevel data in the allocated VSAM data set.

System action

RMF waits for the next input.

User response

Press END (PF3) to return to the previous panel.

GPM0633I **No data: EADM activity data is currently not available.**

Explanation

Within the current report interval, RMF can not report any extended asynchronous data mover (EADM) activity. This could be caused by any of the following reasons:

- EADM activity data was not collected (Monitor III Gatherer option NOEADM was specified).
- EADM is not supported by the processor.

System action

None.

System programmer response

Start the Monitor III gatherer with the EADM option.

IQP messages

This topic describes the MVS messages that are affected by Integrated Accelerator for zEDC.

IQP062I **REQUEST REJECTED - *reasontext***

Explanation

A DISPLAY PCIE, SET IQP or DISPLAY IQP command request was rejected.

In the message text:

reasontext

The reason that the command was rejected can be one of the following:

NOT SUPPORTED BY THE HARDWARE

The z/PCI facility is not supported by the hardware. This command is only allowed when the hardware support is available.

FUNCTION NOT CURRENTLY AVAILABLE

The PCIE and FPGHWAM address spaces either have not completed their initialization

or have terminated. Both of these address spaces must be active in order to process the command. These address spaces are started automatically when the system is IPLed.

OPTIONS IGNORED

The current processor does not use or support the SET IQP options. Therefore, the options are ignored.

System action

The system continues processing.

Operator response

If the reason text indicates that the function is not currently available, reissue the command after the address spaces have completed initialization. If any of

the address spaces have terminated, an IPL is required to restart them.

Source

PCI Express

Module

IQPODPRM, IQPODSP, IQPOTPRM

Descriptor code

5

IQP066I *hh.mm.ss* **DISPLAY IQP**
 text

Explanation

Where *text* is:

```
zEDC Information
MAXSEGMENTS:      maxseg      (ssssM)
Previous MAXSEGMENTS: pmaxseg   (ssssM)
Allocated segments: allocseg    (ssssM)
Used segments:     usedseg     (ssssM)
MINREQSIZE:        minreqszK
DEFMINREQSIZE:      minreqszD (STATIC)
INFMINREQSIZE:      minreqszI (STATIC)
Feature Enablement: featenab
```

This message is issued in response to a DISPLAY IQP command.

In the message text:

maxsegm

The maximum number of 16 MB segments that are currently allowed for problem state zEDC requests as specified by the MAXSEGMENTS keyword in the IQPPRMxx parmlib member.

Note: If a SET IQP command is issued to change the MAXSEGMENTS value to a lower value, the original value remains in effect and is displayed because the maximum number of segments cannot be decreased dynamically. If a SET IQP command is issued to change the MAXSEGMENTS value to a higher value, this higher value is displayed.

Note: The MAXSEGMENTS line will not be displayed when on an IBM z15 and above processors.

pmaxsegm

The previous maximum number of segments that were allowed for problem state requests. Note: If a SET IQP command is issued to change the MAXSEGMENTS value to a lower value, it is ignored because the maximum number of segments cannot be decreased dynamically. If a SET IQP

command is issued to change the MAXSEGMENTS value to a higher value, the previous value is displayed on this line.

allocsegm

The number of segments that have been allocated (page fixed) for problem state zEDC requests.

usedsegm

The number of segments that are in use by problem state zEDC requests.

minreqszD

The minimum request size in kilo/megabytes that is eligible for zEDC compression.

(STATIC) portion of the message

indicates that the value is set during initialization and is no longer a tunable option in the IQPPRMxx parmlib member.

Note: The (STATIC) option will only exist on IBM z15 and above processors, when the zEDC product feature is Enabled.

minreqszI

The minimum request size in kilo/megabytes that is eligible for zEDC compression.

(STATIC) portion of the message

indicates that the value is set during initialization and is no longer a tunable option in the IQPPRMxx parmlib member.

Note: The (STATIC) option will only exist on IBM z15 and above processors, when the zEDC product feature is Enabled.

minreqszK

The minimum request size in kilobytes that is eligible for zEDC compression and decompression as specified on the MINREQSIZE keyword in the IQPPRMxx parmlib member.

featenab

The status of the zEDC product feature. It can be one of the following:

Enabled

The zEDC product feature is enabled (STATE(ENABLED) was specified in IFAPRDxx). zEDC devices are allowed to be used.

Disabled

The zEDC product feature is disabled (STATE(DISABLED) was specified in IFAPRDxx). zEDC devices are not allowed to be used.

None

The zEDC product feature was not defined in IFAPRDxx. zEDC devices are not allowed to be used.

ssssM

The amount of storage in megabytes.

System action

The system continues processing.

Source

PCI Express

Module

IQPODPRM

Descriptor code

5

Appendix A. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation
Site Counsel
2455 South Road*

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

A

- accessibility
 - contact IBM [77](#)
- assistive technologies [77](#)
- authorized interfaces for zEDC [5](#), [53](#), [67](#)

C

- commands
 - DFSMSHsm commands
 - DEFINE [12](#)
 - SETSYS [9](#)
- common address space work
 - data set
 - DD DATA [14](#)
 - DD* [14](#)
 - record type [30](#) [14](#)
- compaction [7](#)
- COMPRESS parameter in IGDSMSxx [8](#)
- compressing file system data
 - always compressed [32](#)
 - displaying status [33](#)
 - monitoring status [33](#)
 - process [31](#)
- compression service
 - memory registration [57](#)
 - Rendezvous [53](#), [56](#)
 - single compression request [61](#)
 - unregister memory [60](#)
 - unrendezvous [66](#)
- contact
 - z/OS [77](#)

D

- data compression [3–5](#)
- data set
 - physical sequential (PS) [7](#)
- DEFINE command
 - for DFSMSHsm [12](#)
 - optional parameters
 - DUMPClass ZCOMPRESS [13](#)
- DFSMSHsm
 - commands
 - DEFINE [12](#)
- DFSMSHsm commands
 - DEFINE [12](#)

E

- EADM (Extended Asynchronous Data Mover Activity Report)
 - Monitor III report
 - command [19](#)
 - contents of [19](#)
 - how to request [19](#)

- EADM (Extended Asynchronous Data Mover Activity Report) (*continued*)
 - Monitor III report (*continued*)
 - purpose of [19](#)
 - Postprocessor report
 - command [21](#)
 - contents of [22](#)
 - how to request [21](#)
 - purpose of [21](#)
- EADM (Extended Asynchronous Data Mover)
 - Monitor III data gatherer session
 - description [19](#)
- EADM|NOEADM (extended asynchronous data mover)
 - Postprocessor [19](#)
- encrypting file system data
 - defining new file systems [32](#)
- ERB629I [73](#)
- ERBSCMG3 (Storage class memory data)
 - format [68](#)
- ERBSCMT3 (tabular report data table) [70](#)
- EXCEPT (exceptional value) control statement
 - Postprocessor
 - I/O queuing activity (type 78-3 SMF record) [24](#)
- Extended Asynchronous Data Mover (EADM) Activity Report (EADM)
 - Postprocessor report
 - command [21](#)
 - how to request [21](#)
- extended asynchronous data mover (EADM) device (subchannel) information section
 - RMF - record type 74-10 [24](#)
- Extended asynchronous data mover (EADM) statistics
 - RMF - record type 74-10 [23](#)
- Extended Asynchronous Data Mover Activity Report (EADM)
 - Monitor III report
 - command [19](#)
 - contents of [19](#)
 - how to request [19](#)
 - purpose of [19](#)
 - Postprocessor report
 - contents of [22](#)
 - purpose of [21](#)
- extended asynchronous data mover reporting (EADM|NOEADM)
 - Postprocessor [19](#)

F

- FPZ4ABC [61](#)
- FPZ4DMR [60](#)
- FPZ4PRB [56](#)
- FPZ4RMR [57](#)
- FPZ4RZV [53](#)
- FPZ4URZ [66](#)

G

GPM0633I [73](#)

H

hardware
 statistics [36](#)

I

I/O queuing activity
 RMF - record type [78-3 27](#)
 SMF record type [78-3](#)
 overview/exception control statements
 [24](#)
IEFACTRT
 record type [30 14](#)
 SMF job/job step termination exit [14](#)
IQP PCIE-related parameters
 displaying [35](#)

K

keyboard
 navigation [77](#)
 PF keys [77](#)
 shortcut keys [77](#)

M

Monitor III data gatherer session
 session option
 EADM (Extended Asynchronous Data Mover) [19](#)

N

navigation
 keyboard [77](#)

O

OVW (overview condition) control statement
 Postprocessor
 I/O queuing activity (type [78-3 SMF record](#))
 [24](#)

P

parameters
 ALL [10](#)
 DASDBACKUP [10](#)
 DASDMIGRATE [11](#)
 NONE [10](#)
 TAPEBACKUP [11](#)
 TAPEMIGRATE [12](#)
 ZCOMPRESS [13](#)
physical sequential data set [7](#)
Postprocessor
 I/O queuing activity (type [78-3 SMF record](#))
 [24](#)

R

RMF - record type [74-10 23](#)
RMF (Resource Measurement Facility)
 Type [74-10](#) - extended asynchronous data mover
 (EADM) device (subchannel) information section [24](#)
 Type [74-10](#) - Storage class memory (SCM) statistics [23](#)
 Type [78-3](#) - I/O queuing activity [27](#)

S

SCM [19](#)
 See also EADM
SETSYS command
 optional parameters
 NONE [10](#)
 ZCOMPRESS ALL [10](#)
 ZCOMPRESS DASDBACKUP [10](#)
 ZCOMPRESS DASDMIGRATE [11](#)
 ZCOMPRESS TAPEBACKUP [11](#)
 ZCOMPRESS TAPEMIGRATE [12](#)
 overview [9](#)
 shortcut keys [77](#)
SMF record
 type [78-3](#) (I/O queuing activity)
 overview/exception control statements
 [24](#)
 solution_name
 what is [1](#)
 storage class memory [19](#)
 See also extended asynchronous data mover (EADM)
 summary of changes [xiii](#)

T

tabular report
 data tables
 ERBSCMT3 [70](#)
trademarks [82](#)

U

user interface
 ISPF [77](#)
 TSO/E [77](#)

Z

zEDC [3–5, 68](#)
zEDC Express
 used in compressing files [31](#)
zEnterprise Data Compression (zEDC)
 requirements [4](#)
zlib for zEDC [5, 47, 50](#)



Product Number: 5655-ZOS