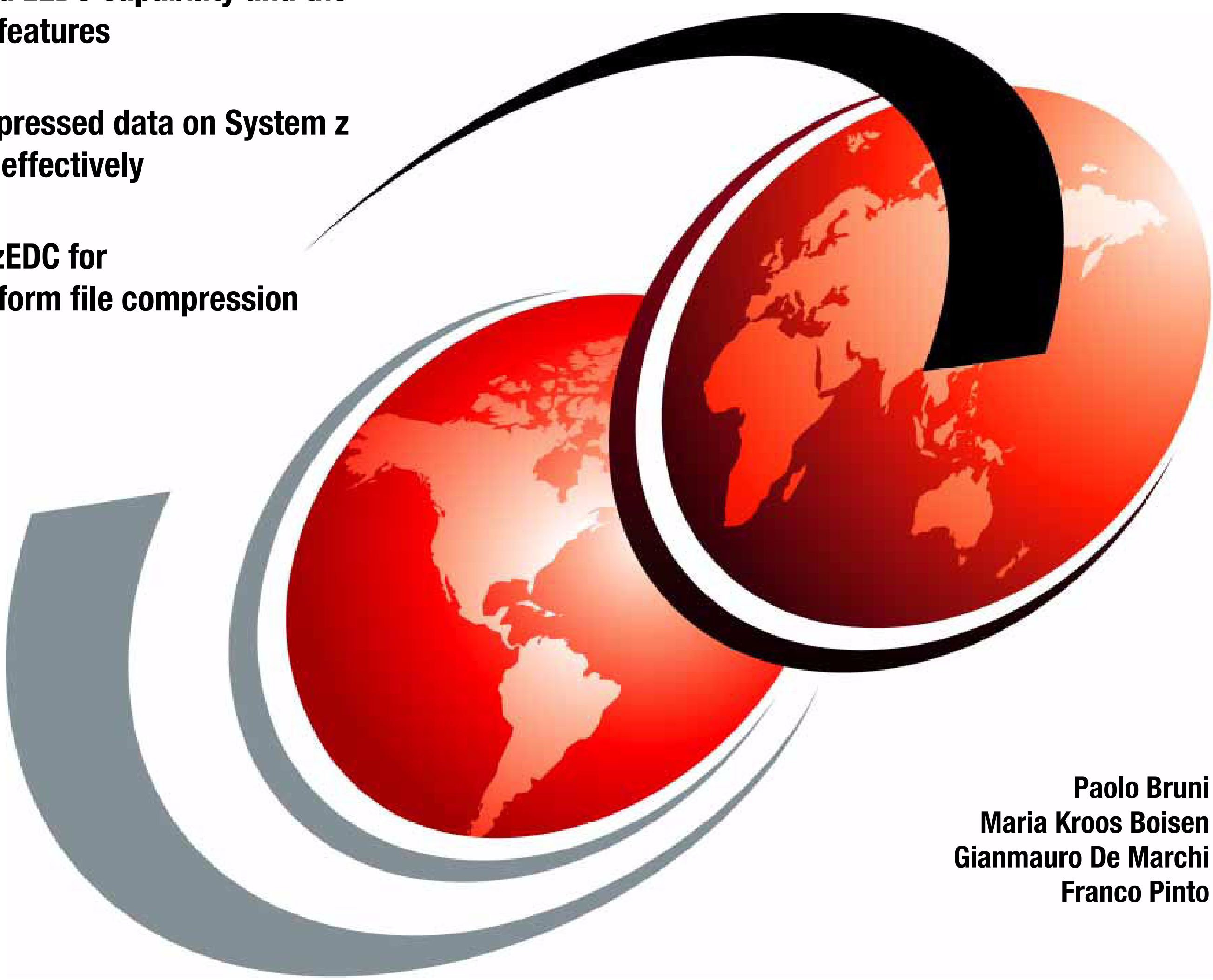
Reduce Storage Occupancy and Increase Operations Efficiency with IBM zEnterprise Data Compression

Understand zEDC capability and the hardware features

Store compressed data on System z more cost effectively

Leverage zEDC for cross-platform file compression



Paolo Bruni
Maria Kroos Boisen
Gianmauro De Marchi
Franco Pinto

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM zEnterprise Data Compression

February 2015

SG24-8259-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.

First Edition (February 2015)

This edition applies to zEnterprise Data Compression, a combination of the z/OS V2.1 zEDC capability and the hardware feature zEDC Express (FC# 0420) available for zEC12 GA2, zBC12 and later models.

© Copyright International Business Machines Corporation 2015. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2015. All rights reserved.

Figures

© Copyright IBM Corp. 2015. All rights reserved.

Tables

© Copyright IBM Corp. 2015. All rights reserved.

Examples

© Copyright IBM Corp. 2015. All rights reserved.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features described in this document in other countries. Consult your local IBM representative for information about 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, Armonk, NY 10504-1785 U.S.A.

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.

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.

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.

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.

© Copyright IBM Corp. 2015. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2®	IMS™	RMF™
DS8000®	MVS™	System z®
eServer™	Parallel Sysplex®	WebSphere®
FICON®	RACF®	z/OS®
FlashCopy®	Redbooks®	z/VM®
Global Technology Services®	Redbooks (logo) ®	zEnterprise®
IBM®	Resource Measurement Facility™	zSecure™

The following terms are trademarks of other companies:

Connect:Direct, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, 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.

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

Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- Search, bookmark, save and organize favorites
- Get up-to-the-minute Redbooks news and announcements
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android

Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

IBM® zEnterprise® Data Compression (zEDC) capability and the Peripheral Component Interconnect Express (PCIe or PCI Express) hardware adapter called *zEDC Express* were announced in July 2013 as enhancements to the IBM z/OS® V2.1 operating system (OS) and the IBM zEnterprise EC12 (zEC12) and the IBM zEnterprise BC12 (zBC12).

zEDC is optimized for use with large sequential files, and uses an industry-standard compression library. zEDC can help to improve disk usage and optimize cross-platform exchange of data with minimal effect on processor usage.

The first candidate for such compression was the System Management Facility (SMF), and support for basic sequential access method (BSAM) and queued sequential access method (QSAM) followed in first quarter 2014. IBM software development kit (SDK) 7 for z/OS Java, IBM Encryption Facility for z/OS, IBM Sterling Connect:Direct® for z/OS and an IBM z/VM® guest can also use zEDC Express.

zEDC can also be used for Data Facility Storage Management Subsystem data set services (DFSMSdss) dumps and restores, and for DFSMS hierarchical storage manager (DFSMShsm) when using DFSMSdss for data moves.

This IBM Redbooks® publication describes how to set up the zEDC functionality to obtain the benefits of portability, reduced storage space, and reduced processor use for large operational sets of data with the most current IBM System z® environment.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Paolo Bruni is an ITSO Project Leader based at the Silicon Valley Lab in San Jose, CA. Since 1998, Paolo has authored IBM Redbooks publications about IBM Information Management System (IBM IMS™), IBM DB2® for z/OS, and related tools, and has conducted workshops worldwide. During his many years with IBM in development and in the field, Paolo's work has been related to System z.

Maria Kroos Boisen is a Client Technical Specialist in IBM Systems and Technology Group (STG) in Denmark. Maria joined IBM in 2008. She graduated from the IT University of Copenhagen in 2011, with a masters degree in Information Technology (IT).

Gianmauro De Marchi is an Advisor IT Specialist in IBM Global Services, Italy. He has 30 years of experience in IBM working with z/OS-based client and systems. His areas of expertise include z/OS, IBM Parallel Sysplex®, and hardware configuration definition (HCD). Gianmauro has supported several projects involving IBM zEnterprise installations for IBM clients in Italy.

Franco Pinto is a Client Technical Specialist in Switzerland. He has 20 years of experience in the mainframe and z/OS field. His areas of expertise include System z technical pre-sales covering mainframe sizing and installation planning, and providing support on existing and new System z functions.

© Copyright IBM Corp. 2015. All rights reserved.

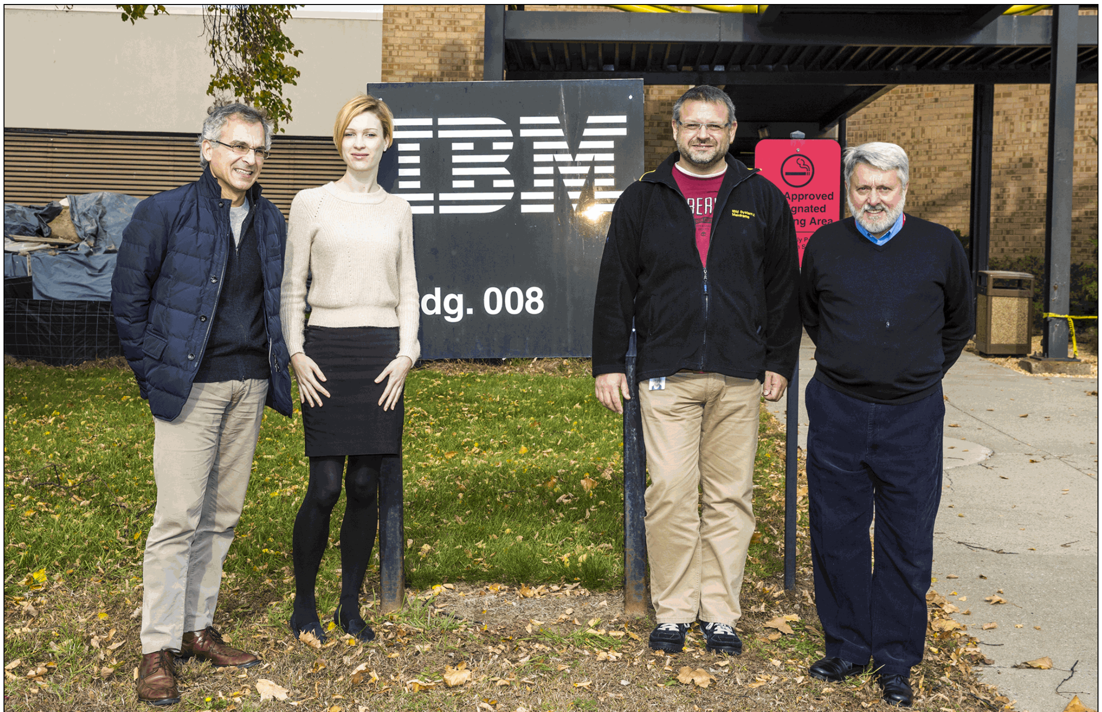


Photo courtesy of Peter Hoyle

The authors in Poughkeepsie. From left to right: Gianmauro, Maria, Franco, and Paolo

Special thanks to Anthony Sofia of IBM STG, who provided technical guidance during the project and made available several of his presentations.

Thanks to the following people for their contributions to this project:
Ann Lund, ITSO

David Betten
John Burg
Michael Groetzner
Steven R. Hart
Cecilia Carranza Lewis
Robert Gensler
Natalia Kutznetsova
Jason Luurs
Alex Patterson
Bertold Reddemann
Peter Sutton
Glenn Wilcock
IBM STG

Bob Haimowitz
IBM Development Support Team

Felipe Bortoletto
IBM Brazil

Gert Laumann
IBM Global Technology Services® (GTS) Delivery, Denmark

Jeannie Vangsness
DFSMShsm Technical Support

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run two - six weeks, and you can participate either in person or as a remote resident working from your home base.

Learn more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form:

  ibm.com/redbooks

- Send your comments in an email:

  redbooks@us.ibm.com

- Mail your comments:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

zEDC overview and prerequisites

In this chapter, we introduce the IBM zEnterprise Data Compression (zEDC) Express for System z, its applicability to use cases, the configuration prerequisites, and some coexistence considerations.

The chapter contains the following sections:

- Overview of zEDC Express
- Use cases
- Configuration

© Copyright IBM Corp. 2015. All rights reserved.

1.1 Overview of zEDC Express

zEDC is a compression acceleration capability, that enables you to do hardware-based data compression and decompression. It is designed for high-performance, low-latency compression, to reduce processor use, optimize the performance of compression-related tasks, improve disk usage, and optimize cross-platform exchange of data.

The solution is a combination of the zEDC capability in IBM z/OS V2.1 and the zEDC Express hardware feature (FC# 0420, see Figure 1-1) available from zEC12 general availability (GA2).

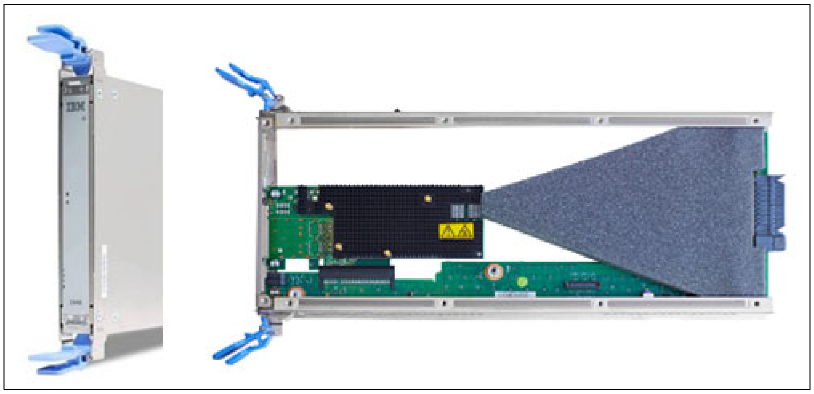


Figure 1-1 The zEDC card (#0420)

The zEDC Express is implemented as Peripheral Component Interconnect Express (PCIe) device. PCIe is a standard for computer expansion cards. It includes a serial bus standard that is used by a large variety of computer platforms.

The input/output (I/O) subsystem direction of IBM System z includes PCIe, InfiniBand, enhanced cards, and other protocols, such as IBM System z IBM Fibre Connection (IBM FICON®) and High Performance FICON for System z (zHPF). It is intended to provide significant performance improvements over the I/O platforms of previous systems both by reducing processor use and latency and providing increased data throughput.

Two examples of PCIe devices are zEDC Express and IBM 10 gigabit Ethernet (GbE) Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) Express.

1.1.1 DEFLATE format

The compression format generated and used by zEDC Express is the Request for Comments (RFC) 1951 DEFLATE format, an industry standard that compresses data using the Lempel-Ziv 1977 (LZ77) and Huffman coding algorithms.

For details about RFC 1951, see the following website:

https://www.ietf.org/rfc/rfc1951.txt

Figure 1-2 shows at a very high level how the zEDC compression works.

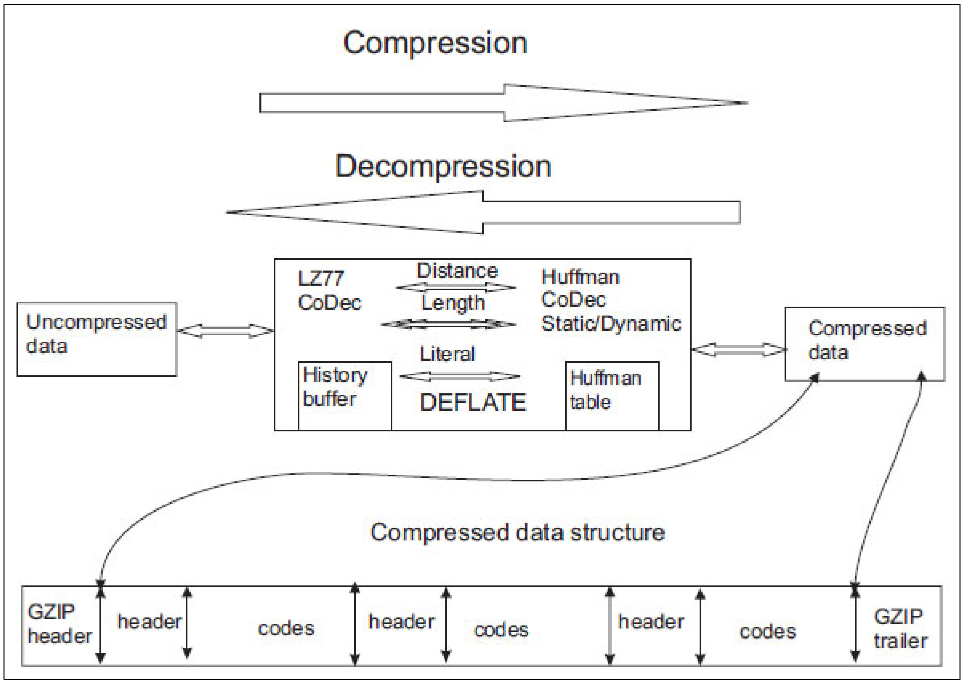


Figure 1-2 Example of zEDC compression

LZ77 provides pattern matching within a 32 kilobyte (KB) rolling window, or look back, in the data. As matches are found, they are replaced with back reference to the match. Huffman coding encodes the symbols in the file into a set of bit patterns, where the most-used symbols get the smallest bit patterns.

The DEFLATE compression is considered dictionary-less, because it hides the dictionary in the data stream. The uncompressed data is the dictionary.

1.1.2 Types of compression

Compression is generally used to reduce auxiliary storage allocation, and provide savings in the following areas:

- Data storage space, by decreasing the size of data that needs to be stored.
- Transmission capacity, by improving I/O throughput (because more compact data needs to be shared or stored).

There are multiple compression technologies offered for System z, of both hardware and software types. The different compression techniques available for compressed format data sets, including generic compression, tailored compression, and zEDC, use different methods to derive a compression dictionary for the data sets:

- Using generic compression, a set of dictionary building blocks (DBBs) in `SYS1.DBBLIB` that best reflects the data in the data set is selected by the system.
- For tailored compression, the dictionary is tailored specifically to the individual data set.
- zEDC hides the dictionary in the data stream, so no separate dictionary needs to be created.

zEDC Express is complementary to the hardware compression on the System z processor chip. The System z processor chip is using the compression call (CMPSC) coprocessor instruction and a dictionary to compress data. The zEDC Express uses the DEFLATE format, and is considered dictionary-less, because it includes the dictionary in the data stream. Hardware compression is optimized for short records, such as database rows, but zEDC is optimized for use with large sequential files.

Table 1-1 shows the compression provided by the coprocessor (starting the hardware-supported CMPSC) in comparison with the zEDC Express (calling zlib to use the DEFLATE compression algorithm).

Table 1-1 Comparison of CMPSC versus zEDC Express

Compression	CMPSC	zEDC Express
Where available	On chip in every IBM eServer™ zSeries today (and tomorrow)	PCIe adapter, new with IBM zEnterprise EC12 (zEC12) GA2, IBM zEnterprise BC12 (zBC12), and later models
Maturity	Decades of use by access methods and IBM DB2	Industry standard with decades of software support
Where run	Work performed jointly by the central processing unit (CPU) and coprocessor	Work performed by the PCIe adapter
Format	Proprietary compression format	Standards-compliant (RFC1951)
Objects compressed	Rows in a database	► Large sequential data ► Queued sequential access method (QSAM) and basic sequential access method (BSAM) online sequential data ► Objects stored in a database
Standard of data	System z only	Cross platform data exchange
Users	► Virtual Storage Access Method (VSAM) for better disk use ► DB2 for lower memory usage ► The majority of DB2 users currently compress their rows ► DFSMShsm/dss	► QSAM/BSAM for better disk use and batch elapsed time improvements ► IBM System Management Facilities (SMF) for increased availability and online storage reduction ► Java for high throughput standard compression using `java.util.zip` ► Data Facility Storage Management Subsystem hierarchical storage manager (DFSMShsm) and DFSMS data set services (DFSMSdss) ► Encryption Facility for z/OS for better industry data exchange ► IBM Sterling Connect:Direct for z/OS for better throughput and link usage ► Independent software vendor (ISV) support for increased client value

CMPSC instructions are used where hardware compression is best suited, and instructions and files that will benefit from zEDC Express compression are directed to this feature.

The zlib open-source library

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. When zEDC is not appropriate, software-based compression services are used. The zlib data compression library provides in-memory compression and decompression functions, and implements the DEFLATE file format.

The wrapper function in the IBM-provided, zlib-compatible C library determines when zEDC compression is appropriate and, if not, software-based compression services are used.

The zEDC-enabled zlib library is available for z/OS UNIX System Services (z/OS V2.1).

For more information about the zlib wrapper function, see the *z/OS MVS Programming: Callable Services for High-Level Languages*, SA23-1377.

1.2 Use cases

Here is a list of the z/OS functions that currently can use the zEDC Express capabilities:

- SMF logstreams

  For increased availability and online storage reduction.
- QSAM and BSAM (sequential) data sets

  For better disk use and batch elapsed time improvements.
- DFSMSdss/DFSMShsm

  When backing up and restoring data, or migrating and recalling data.
- IBM Java V7.0.0 SR7 and Java V7R1 runtime environment

  For high throughput standard compression with `java.util.zip`.
- IBM Encryption Facility for z/OS V2.1

  For building industry-standard compressed OpenPGP[1] (RFC4880) files.
- IBM Sterling Connect:Direct for z/OS V5.2

  For better throughput and link use.
- IBM WebSphere® MQ for z/OS V8

  For channel message compression.
- zlib

  For application programs that directly use the zEDC with the zlib open source library application programming interfaces (APIs).

We describe several use cases on System z in the following chapters of this document.

[1] OpenPGP: A standard that describes how Pretty Good Privacy (PGP) encryption works so that encrypted messages can be handled by different software implementations. PGP is a trademark belonging to Symantec Corp.

1.3 Configuration

zEDC Express is an optional feature (feature code (FC) #0420) made available starting with zEC12 GA2, and z/OS V2.1. The feature installs exclusively on the PCIe I/O drawer. Up to two zEDC Express features can be installed per PCIe I/O drawer domain.

Note that, if the I/O drawer contains a Flash Express or 10 GbE RoCE feature, only one zEDC feature can be installed on that domain. There is one compression coprocessor per zEDC Express feature. A zEDC Express feature can be shared by up to 15 logical partitions (LPARs).

The zEDC Express feature is defined as part of the I/O configuration using the Hardware Configuration Definition (HCD) program or using an I/O Configuration Program (IOCP).

One to eight features can be installed on the system. You need at least two zEDC Express features for high availability. Four features are highly advised to aid with normal maintenance because this provides continuous availability during concurrent update.

It is suggested that you have the zEDC Express and z/OS V2.1 on any LPAR or server you share files with. Servers without the feature and releases prior to z/OS V2.1 (z/OS V1.12 and z/OS V1.13) have toleration support, and are able to decompress data, but this can be a CPU-intensive task. See 1.3.2, “Coexistence” on page 7.

The installation of zEDC is described in Chapter 2, “Installing IBM zEnterprise Data Compression Express devices” on page 9.

IBM Resource Management Facility (IBM RMF™) support for hardware compression includes IBM System Management Facilities (SMF) Type 74 SubType 9 records and a new Monitor I PCIe Activity report, providing information about compression activity on the system. See 3.7, “zEDC and PCIe monitoring” on page 43.

1.3.1 Prerequisites

To use the zEDC feature, the following prerequisites must be in place:

- zEDC Express hardware requirements
  - zEC12 with driver 15E
  - zBC12 with one coprocessor per PCIe I/O feature
  - IBM zNext
  - zEDC Express feature (FC#0420)

  For availability, it is advised to have a minimum of two zEDC Express features. For best performance, all systems accessing the compressed data should have the zEDC Express feature.

- zEDC software requirements
  - z/OS V2.1
  - zEDC Express software feature enabled in a IFAPRDxx parmlib member

  In case zEDC Express is not installed or is unavailable, software decompression support is available on z/OS V2.1, z/OS V1.13, and z/OS V1.12, with appropriate program temporary fixes (PTFs). For more information, see 1.3.2, “Coexistence” on page 7.

> **Important:** For the full benefit of zEDC Express, zEDC Express features, including z/OS V2.1, should be active on all of the systems that might share compressed-format data sets.

1.3.2 Coexistence

For systems that do not support the zEDC Express feature, but have z/OS V2.1, z/OS V1.13, or z/OS V1.12 installed, it is possible to access a zEDC Express compressed-format data set. In this case, compressed data is read from data sets and decompressed using software algorithms. New data being written is not compressed.

OpenPGP packages can be accessed with any industry-standard tooling.

The following list describes the coexistence requirements:

- z/OS V1.12 and z/OS V1.13 PTFs for authorized program analysis report (APAR) OA41156.

  This PTF is needed for the systems to tolerate the new SMFPRMxx keywords, and to enable the IFASMFDL SOFTINFLATE keyword and software decompression support.
- For z/OS V2.1, any IFASMFDL job needs to specify a region size of 4 megabytes (MB) or greater.

  This is needed because IFASMFDL has been enhanced to use multi-block logstream browsing.

> **Important:** Software decompression is slow, and uses considerable processor resources. Therefore, it is not suggested for production environments.

2

Installing IBM zEnterprise Data Compression Express devices

In this chapter, we describe how to upgrade an existing IBM zEnterprise EC12 (zEC12) to install two new IBM zEnterprise Data Compression (zEDC) Express features needed for zEDC functionality.

The chapter contains the following topics:

- Installation planning
- IBM z/OS: Verify the prerequisites
- z/OS: Enabling the Priced Software Feature
- z/OS: Control the use of Peripheral Component Interconnect Express features
- Hardware configuration definition (HCD): Defining the device
- HCD: Activating the new configuration
- z/OS: Bringing the zEDC Express devices online to z/OS
- z/OS: Managing the zEDC Express devices

© Copyright IBM Corp. 2015. All rights reserved.

2.1 Installation planning

Adapter support for zEDC is provided by Resource Group (RG) code running on the system integrated firmware processor (IFP).

For resilience, there are always two independent RGs on the system, sharing the IFP. Install a minimum of two zEDC features, one feature per RG (Figure 2-1).

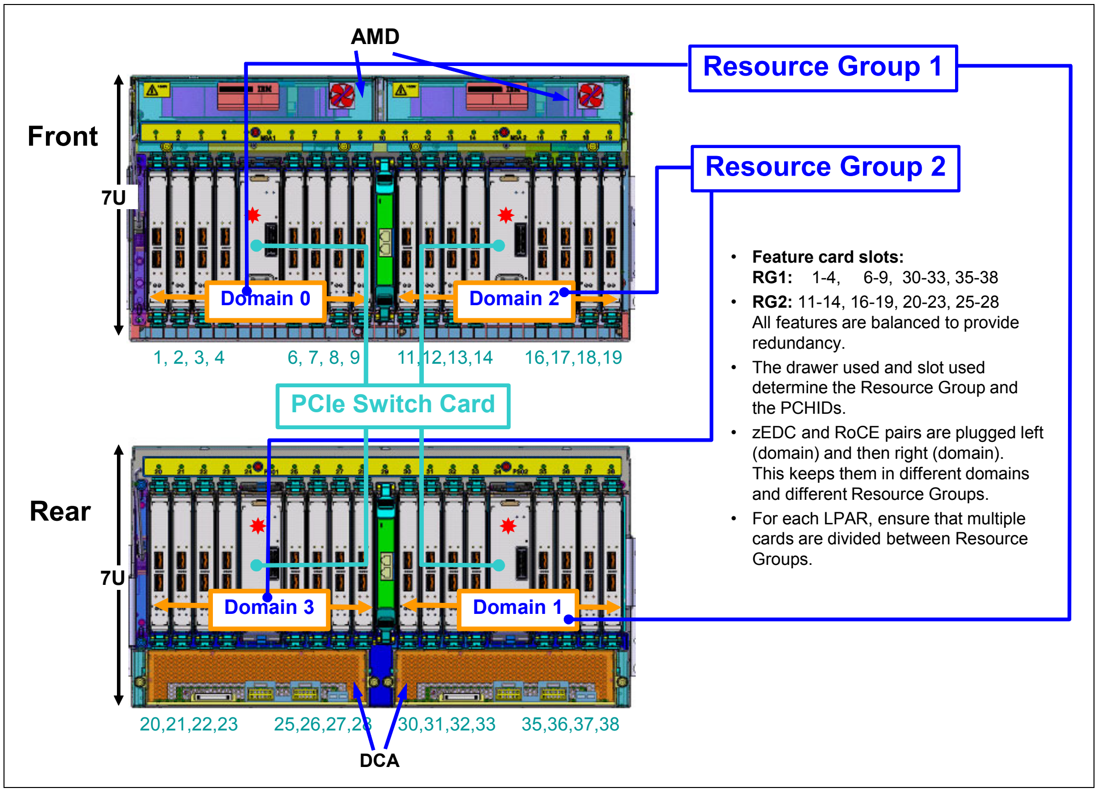


Figure 2-1 Relationship among PCIe I/O cage card slots, I/O domains, and RGs

A zEDC Express feature can be shared by up to 15 logical partitions (LPARs). You need to ensure that an LPAR has access to hardware in both RGs for best availability.

Consider that if one feature becomes unavailable, the other features need to be able to absorb the load. Therefore, for the best data throughput and availability, install at least two features per RG (Figure 2-2).

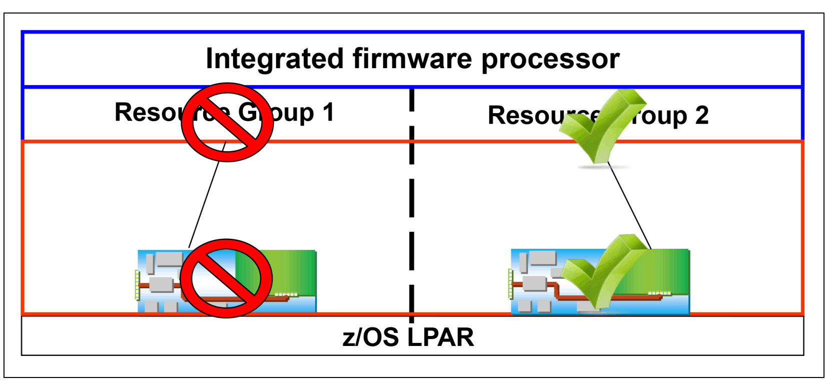


Figure 2-2 IFP and RG basic configuration

For high availability (HA) with minimal effect, especially for zEDC Express, install these native Peripheral Component Interconnect Express (PCIe) features in quantities of four. During general firmware updates, error conditions, and so on, when one RG's features are unavailable, you have a minimum of two native PCIe features available. This configuration prevents a complete loss of that resource (Figure 2-3).

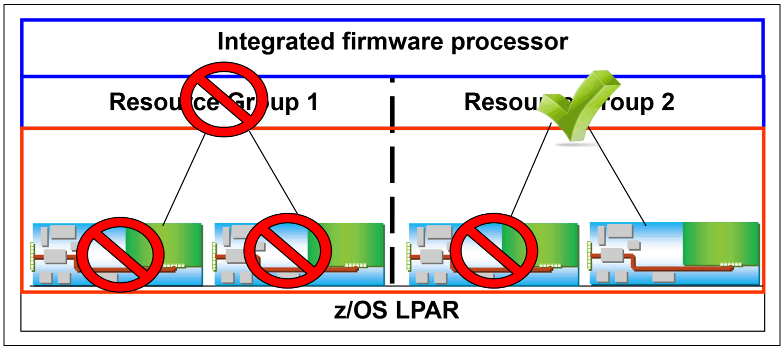


Figure 2-3 IFP and RG advanced configuration

2.1.1 Preconditions

Before starting with definitions, you need to perform the following tasks:

- Plan an initial program load (IPL).
- Obtain a physical channel ID (PCHID).

Plan an IPL

The change described in 2.3, "z/OS: Enabling the Priced Software Feature" on page 14 requires an IPL of z/OS.

> **Note:** zEDC Express device driver recognizes the zEDC Express for z/OS Feature enablement at IPL time.

Obtain PCHID

The PCHID is in the **PCHID REPORT** section of the *Miscellaneous equipment specification (MES) upgrade* documentation.

The key points of the report are listed in Example 2-1.

Example 2-1 MES documentation

```
30179065                      PCHID REPORT                       Oct 28,2014
  Machine: 2827-H43   02000B8D7
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Source           Cage  Slot  F/C    PCHID/Ports or AID            Comment
  06/D2            A25B  D206  0171   AID=09
....
....
  06/DA/J02        Z15B  36    0408   574/J00J01
  06/DA/J02        Z15B  37    0420   578                   RG1     NEW
  15/D9/J01        Z22B  23    0409   5CC/D1 5CD/D2
  15/D9/J01        Z22B  25    0420   5D0                   RG2     NEW
....
....
  06/DA/J01        Z22B  37    0407   5F8/J00
  06/DA/J01        Z22B  38    0865   5FC/P00
```

```
Legend:
  Source  Book Slot/Fanout Slot/Jack
  Z22B    PCIe Drawer 1 in Z frame
  Z15B    PCIe Drawer 2 in Z frame
  Z08B    PCIe Drawer 3 in Z frame
  0408    OSA
  0409    FICON Express8S 10KM LX
  0407    OSA
  RG1     Resource Group One
  RG2     Resource Group Two
  0420    zEDC Express
  0171    HCA3
  0170    HCA3
```

In the example, the key points to be verified are:

FC Feature Code 0420 zEDC Express.

PCHID Physical Channel Identifier to be use on HCD.

RG Resource Group.

2.2 z/OS: Verify the prerequisites

Support of the zEDC Express functionality is provided exclusively by z/OS V2R1 or higher for both data compression and decompression.

Support for data recovery (decompression) in the case that zEDC is not installed, or installed but not available on the system, is provided through software on z/OS V2R1, V1R13, and V1R12 with the appropriate program temporary fixes (PTFs).

Important: By comparison, software decompression is slow and uses considerable processor resources. Therefore, it is not suggested for production environments.

If you are using zlib support, you need to add the `FPZ.ACCELERATOR.COMPRESSION` IBM RACF® profile to protect from unauthorized users of zEDC. To use the zEDC function, you have to install the new function PTFs related to z/OS, Data Facility Storage Management Subsystem hierarchical storage manager (DFSMShsm), DFSMS data set services (DFDSSdss), and Encryption Facility.

An easy way to discover these PTFs is to use the fix category (FIXCAT) IBM System Modification Program/Extended (SMP/E) function. This function is explained on the following website:

http://www.ibm.com/systems/z/os/zos/features/smpe/fix-category.html

There is a specific fix category named IBM.Function.zEDC. This category identifies the fixes that enable or use the zEDC function.

The following fix categories should be also verified:

- The IBM.Device.Server.zBC12-2828.Exploitation

  Fixes that are required to use the capabilities of an IBM zEnterprise BC12 (zBC12) server.
- IBM.Device.Server.zEC12-2827.Exploitation

  Fixes that are required to use the capabilities of an IBM zEnterprise EC12 (zEC12) server.
- IBM.Coexistence.z/OS.V2R1

  Fixes that enable z/OS V1.12 and z/OS V1.13 to coexist with, and fall back from, z/OS V2.1.

The following list describes System z functions that can use zEDC and the related authorized program analysis reports (APARs):

- System Management Facilities (SMF)

  Exploitation APAR OA41817.
- Basic sequential access method (BSAM) and queued sequential access method (QSAM)

  There is a COMPACTION option added to the SMS DATACLAS structure, and a COMPRESS option added to the IGDSMSxx member in parmlib. This additional functionality is delivered with the PTF for APAR OA42195.
- DFSMSdss

  Exploitation APARs: OA42238, OA42198.
- DFSMShsm

  Exploitation APAR: OA42243.
- IBM software development kit (SDK) 7 for z/OS Java, IBM Encryption Facility for z/OS (5655-P97)

  IBM 31-bit and 64-bit SDK for z/OS Java Technology Edition, Version 7 Release 1 (5655-W43 and 5655-W44), which is IBM SDK 7 for z/OS Java, use of zEDC Express. In addition to the PTF for APAR OA43869 and the Java update, the IBM Encryption Facility is also ready to use zEDC Express and zEDC on z/OS v2.1.
- Version 5.2 of IBM Sterling Connect:Direct for z/OS (5655-X01 and 5655-X09)

  Use zEDC Express and z/OS v2.1 zEDC.
- IBM z/VM V6.3 guest on the zEC12 and zBC12 servers.

  This support extends to a z/VM guest only, and not to z/VM directly. Support for a z/VM guest is provided by the PTF for APAR VM65417, plus additional maintenance to a list of components.
- IBM Security zSecure™

  zEDC compression works well with consolidated zSecure Admin Access Monitor data sets.
- IBM WebSphere MQ V8 (IBM MQ)

  The COMPMSG(ZLIBFAST) attribute now uses zEDC, when available, to perform compression and decompression of message data.

APARs of interest

The APARs contain excellent details in their cover.

There are some APARs that might be of interest if you are planning to use zEDC Express. They either fix issues or add functionality, with all being important to making this functionality work successfully.

- OA41245: NEW FUNCTION APAR IN SUPPORT OF ZEDC EXPRESS

  This APAR supplies decompression services compatible with the z/OS authorized interface for zEDC Express.

- OA41156: NEW FUNCTION

  Tolerate SMF zEDC use and SMFPRMxx keywords IFASMFDL. The logstream dump utility is enhanced to decompress any SMF records read when those records were compressed using zEDC.

- OA41817: SMF zEDC exploitation corrections

  Miscellaneous fixes for the zEDC exploitation.

- OA43869: NEW FUNCTION - CRYPTION FACILITY OPENPGP SUPPORT FOR ZEDC

  zEDC is used for compression when a zEDC feature is available on the system, and when using IBM 31-bit SDK for z/OS, Java Technology Edition, Version 7 Release 1 or later.

- OA42196: NEW FUNCTION

  Delivers Extended Format BSAM and QSAM data set compression.

- II14740: HINTS AND TIPS FOR ZEDC USAGE ON ZO/S

  Provides additional guidance for the zEDC implementation on z/OS and z/VM.

> **Note:** New function APAR OA45767, currently open, adds zEDC usage statistics into the SMF30 record.

2.3 z/OS: Enabling the Priced Software Feature

The zEDC Express software support is a priced feature of z/OS.

If you have products that require product enablement, the IFAPRDxx PARMLIB member contains their definitions. zEDC is one of these products.

Specify the following subparameter values:

- NAME of the product it belongs to (in this case, z/OS).
- FEATURENAME (ZEDC).
- ID of the product (5650-ZOS).
- The following optional subparameters all default to asterisk (*):
  - VERSION
  - RELEASE
  - MOD
- STATE (the most important subparameter). STATE can be set to one of the following values:
  - ENABLED
  - DISABLED
  - NOTDEFINED

You are required to change the IFAPRDxx PARMLIB member to include these statements, as shown in Example 2-2.

Example 2-2   IFAPRD03 PARMLIB member

```
PRODUCT OWNER('IBM CORP')
NAME('Z/OS')
ID(5650-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 using **SET PROD=03** IBM MVS™ System Command, you see the output shown in Example 2-3.

Example 2-3   SET PROD command output

```
IFA100I IN PARMLIB MEMBER=IFAPRD03 ON LINE 44
PRODUCTS WITH OWNER=IBM CORP NAME=Z/OS
FEATURE=ZEDC VERSION=*.*.* ID=5650-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 output shown in Example 2-4.

Example 2-4   Feature enablement: Disabled

```
D IQP
IQP066I 14.17.39 DISPLAY IQP
zEDC Information
 MAXSEGMENTS:              N/A
 Previous MAXSEGMENTS:     N/A
 Allocated segments:       N/A
 Used segments:            N/A
 DEFMINREQSIZE:            N/A
 INFMINREQSIZE:            N/A
 Feature Enablement:   Disabled
```

After an IPL, the same command shows the zEDC product feature Enabled (Example 2-5).

Example 2-5   Feature enablement: Enabled

```
D IQP
IQP066I 14.29.51 DISPLAY IQP
zEDC Information
 MAXSEGMENTS:                4  (64M)
 Previous MAXSEGMENTS:     N/A
 Allocated segments:         0  (0M)
 Used segments:              0  (0M)
 DEFMINREQSIZE:              4K
```

```
INFMINREQSIZE:              16K
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 output from this command is shown in Example 2-6.

Example 2-6 D PROD command

```
IFA111I 16.25.49 PROD DISPLAY
S OWNER          NAME            FEATURE          VERSION  ID
E IBM CORP       z/OS            ZEDC             02.01.00 5650-ZOS
```

Additional information about the IFAPRDxx PARMLIB member can be found in the chapter about IFAPRDxx (Product Enablement Policy) of the *z/OS MVS Initialization and Tuning Reference,* SA23-1380.

2.4 z/OS: Control the use of Peripheral Component Interconnect Express features

Assuming that this is your first time using zEDC, the IQPPRMxx member of parmlib has the parameters that control the use of PCIe features, in this case zEDC Express, adjusting internal settings for zlib behavior.

This is a new z/OS V2.1 member that was added to SYS1.PARMLIB.

The statement in the IQPPRMxx member used to manage application requests that use zEDC features is `ZEDC`. zEDC Express is the only feature controlled by this member so far.

The new IQPPRMxx ZEDC parameter has three subparameters that apply to zlib and zlib users only:

MAXSEGMENTS The number of 16 megabytes (MB) storage segments allowed. This value can be increased using the **SET IQP=(xx)** command, but it cannot be lowered. Its default value is `4`.

DEFMINREQSIZE The minimum data size that can be compressed by the zEDC feature specified in kilobytes (KB). Its default size is 4 KB.

INFMINREQSIZE The minimum data size that can be decompressed by the zEDC feature, again specified in KB. The default value is 16 KB.

The IQPPRMxx PARMLIB member for our environment is listed in Example 2-7.

Example 2-7 IQPPRMxx PARMLIB member

```
ZEDC,
  MAXSEGMENTS=4,
  DEFMINREQSIZE=4,
  INFMINREQSIZE=16
```

To verify the allocated and used 16 MB storage segments, you can use the **DISPLAY IQP** MVS command. The output is shown in Example 2-8.

Example 2-8 Display IQP command

```
IQP066I 17.23.34 DISPLAY IQP
zEDC Information
 MAXSEGMENTS:                4  (64M)
 Previous MAXSEGMENTS:     N/A
 Allocated segments:         1  (16M)
 Used segments:              0  (0M)
 DEFMINREQSIZE:              4K
 INFMINREQSIZE:             16K
 Feature Enablement:   Enabled
```

Additional information about the IQPPRMxx PARMLIB member can be found in the chapter about IQPPRMxx (PCIe related parameters) in the *z/OS MVS Initialization and Tuning Reference,* SA23-1380.

2.5 HCD: Defining the device

This section describes the setup and its verification.

There are several differences between the new PCIe adapters (Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) adapter, zEDC-Express adapter) and ordinary channel-path identifiers (CHPIDs).

The PCIe adapters are adapters that have a PCHID and you assign them to an LPAR. But you don't need to assign them to a channel subsystem, and you don't need control units or devices. PCIe adapters provide PCIe functions for an LPAR. These functions are identified by a function ID (FID). For zEC12 and zBC12, a FID is a number in the range `00` - `FF`. The FID must be unique for your server. These two PCIe adapter types have different attributes and uses, as follows:

- The RoCE adapter is great for high-throughput, low-latency communication. It has an FID, PCHID, and physical network (PNET) IDs.
- The zEDC-Express adapter is better for offloading data compression from the processor. It has no PNET IDs, can be virtualized, and holds up to 15 different Virtual Function IDs (VFIDs) in the range 1 - 15.

Similar to reconfigurable CHPIDs, a PCIe function can only be operated by one LPAR at a time, so you must define a PCIe function to an LPAR. The PCIe function can have only one LPAR in its access list, but up to 15 LPARs in its candidate list. Because a PCIe adapter is not accessed through a channel subsystem, you can choose any LPAR of any channel subsystem. As with all objects in HCD, a PCIe function has a description field.

Unlike CHPIDs, the PCHID value is a required input field when you add a PCIe adapter. (For a CHPID, you have the option not to specify the PCHID value when you add the adapter. You can assign that value later by using the CHPID mapping tool.)

To help you monitor PCHIDs that are used in the input/output (I/O) definition file (IODF), HCD provides a new report. This PCHID report provides information about which PCHIDs are already used by your processor, which type of adapter uses the PCHID values and, if available, which PNET IDs are used by that adapter. For PCIe adapters, the report also contains information about the FIDs and, if applicable, the VF IDs used by that adapter.

2.5.1 HCD PCIe function configuration

The first configuration step consists in defining the zEDC Express functions to z/OS with an IODF update using HCD.

You need to complete the following steps:

1. From the HCD main menu, select `Option 1.3` Processor List (Figure 2-4) and type `f` on the processor (in this example, SCZP401).

```
                               Processor List        Row 1 of 3 More:
Command ===> _______________________________________________ Scroll ===> PAGE

Select one or more processors, then press Enter. To add, use F11.


/ Proc. ID Type +   Model +  Mode+ Serial-# + Description
_ SCZP201  2097     E26      LPAR  01DE502097 Eclipse
f SCZP401  2827     H43      LPAR  00B8D72827 Helix
```

Figure 2-4 Processor List: Work with PCIe functions

2. Now you can add the PCIe function (Figure 2-5).

```
----------------------------- Add PCIe Function -----------------------------


Specify or revise the following values.

Processor ID  . . . . : SCZP401      Helix.

Function ID . . . . . . ___
Type  . . . . . . . . . ____________  +

PCHID . . . . . . . . . ___
Virtual Function ID . . __  +

Description . . . . . . ________________________________
```

Figure 2-5 Add PCIe functions: Insert appropriate information

3. Enter the appropriate information. For this example, we specified the following values:

Function ID	`020`
Type	`ZEDC-EXPRESS`
PCHID	`578`
Virtual Function ID	`1`
Description	`LABSERV`

Consider: When you select the value, consider the following items:

- Each PCIe function is identified by a three-digit hexadecimal function ID that is unique within a processor configuration.
- Multiple PCIe functions can be defined for the same PCHID by assigning a unique virtual function number to each of these functions.
- A VFID must not be duplicated on the same PCHID specification.

4. Press Enter. The Define Access List panel opens, showing the list of partitions where you can specify one partition to be connected to the defined PCIe function (Figure 2-6).

```
                              Define Access List
                                                                   Row 1 of 60
 Command ===> _________________________________________ Scroll ===> PAGE

 Select one or more partitions for inclusion in the access list.

Function ID  . . . . : 020

/ CSS ID Partition Name   Number Usage Description
_ 0      A09              9      OS    WtSCPLX8 SC81
_ 1      A1A              A      OS    zAware2
_ 1      A1B              B      CF/OS CHPID holder
_ 1      A1C              C      OS    CLOUD2
_ 1      A1D              D      CF    Trainer FACIL06
_ 1      A1E              E      CF    COMMPLEX CF38
_ 1      A1F              F      CF    COMMPLEX CF39
_ 1      A11              1      OS    COMMPLEX SC30
_ 1      A12              2      OS    VMLINUX9
_ 1      A13              3      OS    COMMPLEX SC31
_ 1      A14              4      OS    zAwareLBS
/ 1      A15              5      OS    LABSERV
_ 1      A16              6      OS    COMMPLEX SC32
_ 1      A17              7      OS    VMLINUX4
```

Figure 2-6   Define Access List: Specify one partition

Restriction: If you select more than one partition, you receive a message indicating that a PCIe function can have only one partition in its access list.

5. Press Enter. The Define Candidate List panel opens. Here you can assign the candidate partitions to the PCIe function (Figure 2-7).

```
                          Define Candidate List
                                                                Row 1 of 59
Command ===> ________________________________________ Scroll ===> PAGE

Select one or more partitions for inclusion in the candidate list.

Function ID  . . . . : 020

/ CSS ID Partition Name   Number Usage Description
_ 0      A0A              A      OS    ITSOZVM3
_ 0      A0B              B      OS    CHPID holder
_ 0      A0C              C      CF    WTSCPLX8 CF8A
_ 0      A0D              D      CF    WTSCPLX8 CF8B
_ 0      A0E              E      CF    TESTPLEX CF7A
_ 0      A0F              F      CF    WTSCPLX1 CF02
_ 0      A01              1      OS    TESTPLEX SC74
_ 0      A02              2      OS    VMLINUX5
_ 0      A03              3      OS    SC76
_ 0      A04              4      OS    VMLINUX7
_ 0      A05              5      OS    ITSOZVM1
_ 0      A06              6      OS    CLOUD1
_ 0      A07              7      OS    VMLINUX8
_ 0      A08              8      OS    WTSCPLX8 SC80
```

Figure 2-7 Define Candidate List: Specify one partition

6. Press Enter.
7. Repeat steps 2 - 6, changing the FID and the VFID for each partition that must share the same zEDC feature.

   In our example, we maintain the same VFID for the same partition, but we assign a different FID.

2.5.2 HCD PCIe function verification

To review the definitions, a report can be produced as follows:

1. From the HCD main menu, select `Option 3.1. Print Configuration Report`. Select the Cascading Style Sheets report (CSS report) by typing a Forward slash (/), and press Enter (Figure 2-8).

```
                         Print Configuration Reports


Select the types of report you want, and specify the values below.

IODF name  : 'SYS6.IODF44.WORK'

Types of report                  Limit report(s)
/  CSS report                    1  1. Yes
_  Switch report                    2. No
_  OS report
_  CTC connection report
_  I/O path report

Job statement information
//*        JOB (ACCOUNT),'NAME'
//*
//*
//*
//*
//*
```

Figure 2-8 Print Configuration Report: CSS report

2. HCD displays the Available CSS Report Types panel. Select CSS summary reports using a Forward slash (/) and pressing Enter (Figure 2-9).

```
        Available CSS Report Types


 Select one or more.

 /  CSS summary reports
 _  Channel path detail reports
 _  Control unit detail report
 _  Device detail report
```

Figure 2-9 Available CSS Report Types: CSS summary reports

3. HCD displays the Limit Reports panel, limited in Figure 2-10 to Processor ID SCZP401.

```
                              Limit Reports


 To limit the reports, specify the following criteria related
 to the IODF in access.



                                      Applicable for:
 Processor ID . . . . SCZP401   +  CSS, CTC, I/O path reports
 Partition name . . . ________  +  CSS, CTC, I/O path report
 OS configuration ID  ________  +  OS, I/O path report
 Switch ID  . . . . . __        +  switch report

 Specify the sysplex and system name to gather the actual
 configuration from. (Blanks default to the local system.)

 Sysplex name . . . . ________     I/O path report
 System name  . . . . ________     I/O path report
```

Figure 2-10 Limit Reports

4. Press Enter. HCD submits a batch job to create a report. See Example 2-9.

Example 2-9 PCIe function summary report

```
PCIe FUNCTION SUMMARY REPORT                  TIME: 15:28 DATE:
 PROCESSOR ID  SCZP401  TYPE  2827     MODEL  H43     CONFIGURATION MODE: LPAR
                                               PARTITION NUMBERS
                          CSS0----------- CSS1----------- CSS2----------- CSS3-----------
FID VF PCHID TYPE         123456789ABCDEF 123456789ABCDEF 123456789ABCDEF 123456789ABCDEF DESCRIPTION

___ __ _____ ____________ _______________________________________________________________ ___________
020  1 578   ZEDC-EXPRESS --------------- ----A---------- --------------- ---------------  LABSERV
021  2 578   ZEDC-EXPRESS --A------------ --------------- --------------- ---------------  SC76
022  3 578   ZEDC-EXPRESS --------------- --------A------ --------------- ---------------  #@$A
023  4 578   ZEDC-EXPRESS --------------- --------------- --------------- A--------------  SC63
024  5 578   ZEDC-EXPRESS --------------- --------------- --------------- -A-------------  SC64
025  6 578   ZEDC-EXPRESS --------------- --------------- --------------- --A------------  SC65
026  7 578   ZEDC-EXPRESS --------------- --------------- --------------- ---A-----------  SC70
027  8 578   ZEDC-EXPRESS A-------------- --------------- --------------- ---------------  SC74
028  9 578   ZEDC-EXPRESS --------------- --------------- --------------- --------A------  SC75
029 10 578   ZEDC-EXPRESS -------A------- --------------- --------------- ---------------  SC80
02A 11 578   ZEDC-EXPRESS --------A------ --------------- --------------- ---------------  SC81
02B 12 578   ZEDC-EXPRESS --------------- --------------- ------A-------- ---------------  SC61
02C 13 578   ZEDC-EXPRESS --------------- --------------- --------------- ---------A-----  SC62
030  1 5D0   ZEDC-EXPRESS --------------- ----A---------- --------------- ---------------  LABSERV
031  2 5D0   ZEDC-EXPRESS --A------------ --------------- --------------- ---------------  SC76
032  3 5D0   ZEDC-EXPRESS --------------- --------A------ --------------- ---------------  #@$A
033  4 5D0   ZEDC-EXPRESS --------------- --------------- --------------- A--------------  SC63
034  5 5D0   ZEDC-EXPRESS --------------- --------------- --------------- -A-------------  SC64
035  6 5D0   ZEDC-EXPRESS --------------- --------------- --------------- --A------------  SC65
036  7 5D0   ZEDC-EXPRESS --------------- --------------- --------------- ---A-----------  SC70
037  8 5D0   ZEDC-EXPRESS A-------------- --------------- --------------- ---------------  SC74
038  9 5D0   ZEDC-EXPRESS --------------- --------------- --------------- --------A------  SC75
039 10 5D0   ZEDC-EXPRESS -------A------- --------------- --------------- ---------------  SC80
03A 11 5D0   ZEDC-EXPRESS --------A------ --------------- --------------- ---------------  SC81
03B 12 5D0   ZEDC-EXPRESS --------------- --------------- ------A-------- ---------------  SC61
03C 13 5D0   ZEDC-EXPRESS --------------- --------------- --------------- ---------A-----  SC62
```

2.6 HCD: Activating the new configuration

To use the definitions that were updated in HCD, you create a production IODF from the work IODF.

To create a production IODF, complete the following steps:

1. From the HCD main menu, select option `2. Activate or process configuration data` (Figure 2-11).

```
                       z/OS V2.1 HCD
Command ===> ________________________________________________________________

                           Hardware Configuration

Select one of the following.

2_  0.  Edit profile options and policies
    1.  Define, modify, or view configuration data
    2.  Activate or process configuration data
    3.  Print or compare configuration data
    4.  Create or view graphical configuration report
    5.  Migrate configuration data
    6.  Maintain I/O definition files
    7.  Query supported hardware and installed UIMs
    8.  Getting started with this dialog
    9.  What's new in this release

For options 1 to 5, specify the name of the IODF to be used.

I/O definition file . . . 'SYS6.IODF44.WORK'                +
```

Figure 2-11 Main menu: Select Activate or process configuration data

2. The Activate or Process Configuration Data panel opens (Figure 2-12). `Select Option 1. Build production I/O definition file` and press Enter.

```
------ Activate or Process Configuration Data ------


 Select one of the following tasks.

 1_  1.  Build production I/O definition file
     2.  Build IOCDS
     3.  Build IOCP input data set
     4.  Create JES3 initialization stream data
     5.  View active configuration
     6.  Activate or verify configuration
         dynamically
     7.  Activate configuration sysplex-wide
     8.  *Activate switch configuration
     9.  *Save switch configuration
     10. Build I/O configuration data
     11. Build and manage System z cluster IOCDSs,
         IPL attributes and dynamic I/O changes
     12. Build validated work I/O definition file

 * = requires TSA I/O Operations
```

Figure 2-12 Activate or Process Configuration Data: Select Build production IODF

3. The Message List panel opens (Figure 2-13). Verify that you have at most Severity W warning messages, and that they are normal for your configuration. Correct any messages that should not occur and try to build the production IODF again. Continue this process until you have no messages that indicate problems. Press PF3 to continue.

```
------------------------------- Message List --------------------------------
   Save  Query  Help
 --------------------------------------------------------------------------
                                                                  Row 1 of 3
 Command ===> ___________________________________________ Scroll ===> PAGE

 Messages are sorted by severity. Select one or more, then press Enter.

 / Sev Msg. ID  Message Text
 _ W   CBDG081I Following 3 operating system configurations of type MVS
 #              have no console devices defined: ALLDEV, LABSERV1,
 #              L06RMVS1
 **************************** Bottom of data ******************************
```

Figure 2-13 Message List (building Production IODF)

4. The Build Production I/O Definition File panel opens (Figure 2-14). Complete the `Production IODF name` and `Volume serial number` fields and press Enter.

```
-------------- Build Production I/O Definition File ----------


 Specify the following values, and choose how to continue.

 Work IODF name . . . : 'SYS6.IODF44.WORK'

 Production IODF name . 'SYS6.IODF44'______________________
 Volume serial number . IODFPK  +

 Continue using as current IODF:
 2   1.  The work IODF in use at present
     2.  The new production IODF specified above
```

Figure 2-14 Build Production I/O Definition File

5. The Define Descriptor Fields panel opens (Figure 2-15). Press Enter to accept the descriptor fields selected by HCD, or enter different values, and then press Enter.

```
-------------------- Define Descriptor Fields ------------------


 Specify or revise the following values.

 Production IODF name  . : 'SYS6.IODF44'

 Descriptor field 1  . . . SYS6
 Descriptor field 2  . . . IODF44
```

Figure 2-15 Define Descriptor Fields

6. HCD displays the following message, indicating that the production IODF was successfully created:

```
Production IODF SYS6.IODF44 created.
```

Now you are ready to activate the new configuration as usual in your environment.

By inspecting the SYSLOG, after the HCD activation, you can find the messages shown in Example 2-10.

Example 2-10 Messages PCIe function available

```
IQP034I PCIe FUNCTION 0020 AVAILABLE FOR CONFIGURATION.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).
IQP034I PCIe FUNCTION 0030 AVAILABLE FOR CONFIGURATION.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).
```

The *z/OS Hardware Configuration Definition User's Guide,* SC34-2669 provides information about working with zEDC Express adapters in the chapter about working with PCIe functions.

2.7 z/OS: Bringing the zEDC Express devices online to z/OS

After the device is defined the next step is to make sure z/OS has access to it.

The *z/OS MVS system commands,* SA38-0666 manual provides information about the following options:

DISPLAY PCIe Display PCIe functions and their associated device types.

CONFIG PFID Configure ON/OFF a specified PCIe function identifier (PFID).

The initial status of devices is `STNBY`, as the **`DISPLAY PCIe`** MVS system command output shows in Example 2-11.

Example 2-11 Display PCIe short format

```
D PCIe
IQP022I 14.35.17 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME        STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator    STNBY                  0578  0001
0030  Hardware Accelerator    STNBY                  05D0  0001
```

The `STNBY` status denotes that the device is in standby mode and ready to be configured online. Now you can bring the devices online using **`CONFIG PFID(020),ONLINE`** and **`CONFIG PFID(030),ONLINE`** MVS system commands (Example 2-12).

Example 2-12 Config PFID online

```
CONFIG PFID(020),ONLINE
IQP034I PCIe FUNCTION 0020 ONLINE.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).
IEE504I PFID(20),ONLINE
IEE712I CONFIG   PROCESSING COMPLETE

CONFIG PFID(030),ONLINE
IQP034I PCIe FUNCTION 0030 ONLINE.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).
IEE504I PFID(30),ONLINE
IEE712I CONFIG   PROCESSING COMPLETE
```

Confirmation message IEE504I is displayed, you can also verify the changed status.

A new display of PCIe shows the status changed to `ALLC`. This status indicates that the device is allocated or in use (Example 2-13).

Example 2-13 New status of PCIe short format

```
D PCIe
IQP022I 14.41.36 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME        STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator    ALLC    0013  FPGHWAM  0578  0001
0030  Hardware Accelerator    ALLC    0013  FPGHWAM  05D0  0001
```

More detailed information can be obtained using the **DISPALY PCIe,PFID=020** MVS system command. See the output in Example 2-14.

Example 2-14 New status of PCIe extended format

```
D PCIe,PFID=020
IQP024I 14.41.48 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     ALLC    0013  FPGHWAM  0578  0001
 CLIENT ASIDS: NONE
 Application Description: zEDC Express
 Device State: Ready
 Adapter Info - Relid: 00000B  Arch Level: 03
               Build Date: 02/13/2014  Build Count: 03
 Application Info - Relid: 000000  Arch Level: 02
```

In addition to the status, you can also see two address spaces new for z/OS V2R1:

PCIe ASID 012 PCI Express.

FPGHWAM ASID 013 Hardware Accelerator Manager.

They provide the infrastructure for PCIe I/O and hardware accelerator activities. These address spaces are started automatically during z/OS initialization, if the appropriate z/OS PCIe facilities hardware is installed (that is, if you are running on at least a zEC12 or zBC12). They are persistent address spaces.

If the PCIe address space is successfully initialized, the following message is displayed:

```
IQP002I PCIe INITIALIZATION COMPLETE
```

If the required hardware is not installed (that is, you are not running on at least a zEC12 or zBC12), the following message is written to the hardcopy log:

```
IQP031I REQUESTED SERVICE IS UNSUPPORTED BY HARDWARE
```

For information about the PCIe messages, see *z/OS MVS System Messages, Volume 9 (IGF-IWM),* SA38-0676.

For information about the FPGHWAM (Hardware Accelerator Manager) messages, see *z/OS MVS System Messages, Volume 5 (EDG-GFS),* SA22-7635.

2.8 z/OS: Managing the zEDC Express devices

During firmware updates, *all* of the features attached to that RG are unavailable. Microcode library (MCL) update to a Resource Group requires an RG outage of a few minutes.

You might be required to set the zEDC Express devices offline. To configure PFID 020 offline, you issue the **cf pfid(20),offline** MVS system command. The output is shown in Example 2-15.

Example 2-15 Configure PFID offline CURRENTLY IN USE

```
CONFIG PFID(020),OFFLINE
IEE148I PFID(20) NOT RECONFIGURED - PCI FUNCTION CURRENTLY IN USE
IEE712I CONFIG   PROCESSING COMPLETE
```

A display of the detailed status with **`D PCIe,PFID(020)`** MVS system command shows that FPGHWAM is allocating the device. See Example 2-16.

Example 2-16 D PCIe allocating devices

```
D PCIe,PFID=020
IQP024I 14.34.51 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     ALLC    0013  FPGHWAM  0578  0001
 CLIENT ASIDS: NONE
 Application Description: zEDC Express
 Device State: Ready
 Adapter Info - Relid: 00000B  Arch Level: 03
                Build Date: 02/13/2014  Build Count: 03
 Application Info - Relid: 000000  Arch Level: 02
```

You need to issue the **`cf pfid(20),offline,force`** MVS system command. See Example 2-17.

Example 2-17 Configure PFID offline FORCE

```
CONFIG PFID(020),OFFLINE,FORCE
IEE505I PFID(20),OFFLINE
IEE712I CONFIG   PROCESSING COMPLETE
```

The status of PCIe device becomes STNBY. See Example 2-18.

Example 2-18 PCIe status STNBY

```
D PCIe,PFID=020
IQP024I 15.16.21 DISPLAY PCIe 096
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     STNBY                  0578  0001
 CLIENT ASIDS: NONE
```

All possible status showed by message IQP024I are:

ALLC The device is allocated or in use.

CNFG The device is configured online.

STNBY The device is in standby mode and ready to be configured online.

DP The device is deallocate-pending and is waiting for a deallocate command from its owner to clean up its resources.

PERR The device is in permanent error. It must be unconfigured to recover from this condition.

To set the PCIe device online, you can use the **`CONFIG PFID(020),ONLINE`** MVS system command, as detailed in 2.7, “z/OS: Bringing the zEDC Express devices online to z/OS” on page 26.

PFIDs can also be configured offline and online using the Support Element (SE). z/OS reacts accordingly to the PCIe availability events that are presented. The sequence of messages is shown in Example 2-19.

Example 2-19 Config PCIe from SE (Support Element)

```
D PCIe
IQP022I 15.50.46 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     ALLC    0013  FPGHWAM  0578  0001
0030  Hardware Accelerator     ALLC    0013  FPGHWAM  05D0  0001

→ Toggle OFF FID 0020 from SE (Support Element)

IQP034I PCIe FUNCTION 0020 NOT AVAILABLE FOR USE.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).
IQP034I PCIe FUNCTION 0020 AVAILABLE FOR CONFIGURATION.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).

D PCIe
IQP022I 15.53.24 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     STNBY                  0578  0001
0030  Hardware Accelerator     ALLC    0013  FPGHWAM  05D0  0001

→ Toggle ON FID 0020 from SE (Support Element)

IQP034I PCIe FUNCTION 0020 ONLINE.
 PCIe DEVICE TYPE NAME = (Hardware Accelerator    ).

D PCIe
IQP022I 15.54.12 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     ALLC    0013  FPGHWAM  0578  0001
0030  Hardware Accelerator     ALLC    0013  FPGHWAM  05D0  0001

D PCIe,PFID=020
IQP024I 16.02.31 DISPLAY PCIe
PCIe     0012 ACTIVE
PFID  DEVICE TYPE NAME         STATUS  ASID  JOBNAME  PCHID VFN
0020  Hardware Accelerator     ALLC    0013  FPGHWAM  0578  0001
 CLIENT ASIDS: NONE
 Application Description: zEDC Express
 Device State: Ready
 Adapter Info - Relid: 00000B  Arch Level: 03
               Build Date: 02/13/2014  Build Count: 03
 Application Info - Relid: 000000  Arch Level: 02
```

Because PFID 20 is currently in use, two IQP034I messages are issued:

- The first IQP034I message indicates that PFID 20 is `NOT AVAILABLE FOR USE`, then the termination of the current in-use instance of that PFID.
- The second IQP034I message indicates that the PFID is now `AVAILABLE FOR CONFIGURATION` in the STANDBY (or OFFLINE) status and ready to be configured.

3

z/OS zEnterprise Data Compression and System Management Facilities

The first application targeted for IBM zEnterprise Data Compression (zEDC) made available from IBM has been the compression of System Management Facilities (SMF) data.

This chapter includes the following sections:

- Introduction to SMF use
- SMF use
- Setting up IBM z/OS SMF
- Dumping compressed records
- Coexistence
- Test case
- zEDC and Peripheral Component Interconnect Express (PCIe) monitoring

© Copyright IBM Corp. 2015. All rights reserved.

3.1 Introduction

SMF data can be massive, and its volume is always increasing. There always seems to be someone wanting to use SMF to record something, a new SMF record or record subtype, new or extended fields in existing SMF records, and in the case of IBM DB2, a new or enhanced Instrumentation Facility Component ID (IFCID) written to the SMF 100, 101, or 102. Of course, without SMF we would have no way to know what's going on with z/OS or one its subsystems.

If compressing your SMF data sounds interesting, as it should, zEDC is the answer.

> **Requirement:** To use zEDC to compress your SMF data, you have to use logstreams *only*.
>
> SMF data sets are not applicable to zEDC.
>
> SMF can use either logstreams or SMF data sets, but NOT both.

If you haven't moved to logstreams yet, this might be a good incentive.

Additional detail about SMF and logstreams can be found in *SMF Logstream Mode: Optimizing the New Paradigm*, SG24-7919.

Extra details specifically about the System Logger are in *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898.

In case of DB2 for z/OS, compressing SMF data using zEDC can either take the place of, or be in addition to, the SMF software compression feature available in DB2 (the **`SMFCOMP`** DB2 subsystem parameter set to `ON` in the DSN6SYSP macro provides end-to-end compression for DB2 SMF records).

3.1.1 Benefits

The advantage of zEDC usage on SMF is to alleviate constraints across the entire lifecycle of a record, as shown in Figure 3-1 on page 33.

Compressing SMF logstreams reduces the amount of data in System Logger. zEDC compresses data up to 4x, saving up to 75% of your sequential data disk space, and reduces the elapsed time to extract IFASMFDL data up to 15%.

In a test in Poughkeepsie lab, DB2 SMF records, which had already been compressed by DB2 with SMFCOMP, achieved a further 50% compression with zEDC.

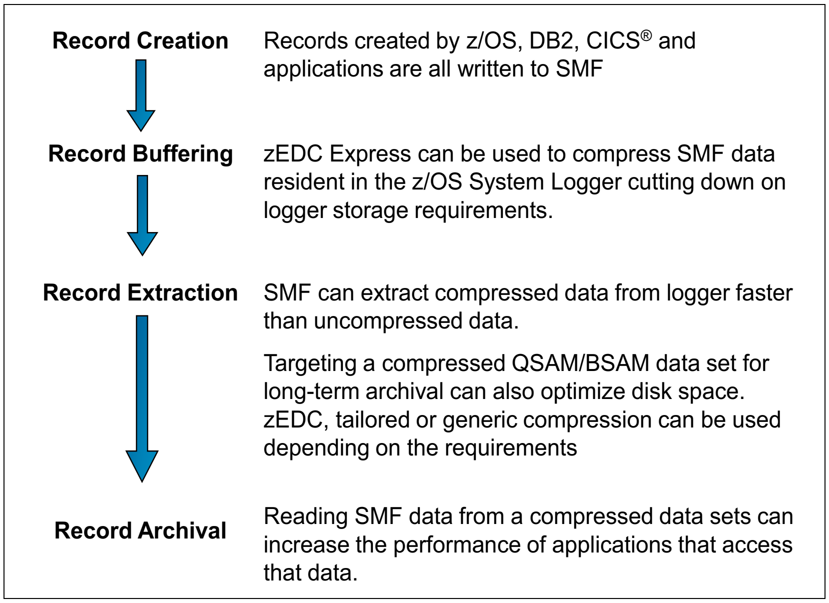


Figure 3-1 zEDC with z/OS SMF logger

3.2 SMF use

In z/OS V2R1, SMF can be configured to use zEDC Express for increased throughput of SMF record logging. This can increase the recording throughput, enabling the following functions:

- Capture of extra SMF data currently uncollected because of System Logger constraints. Coupling facility (CF) and storage management subsystem (SMS) direct access storage device (DASD) are examples of such constraints.
- Mitigation of z/OS image growth because of consolidation, new workloads, or growing workloads, which cause more SMF data to be generated.

> **Restriction:** SMF use of zEDC Express is built on top of logstream recording.
>
> zEDC is not supported for use with SYS1.MANx data sets.

When SMFPRMxx specifies `COMPRESS` on the **`LSNAME`** or **`DEFAULTLSNAME`** parameters, SMF has zEDC Express compress a buffer of SMF records before it is written to the system logger (Figure 3-2).

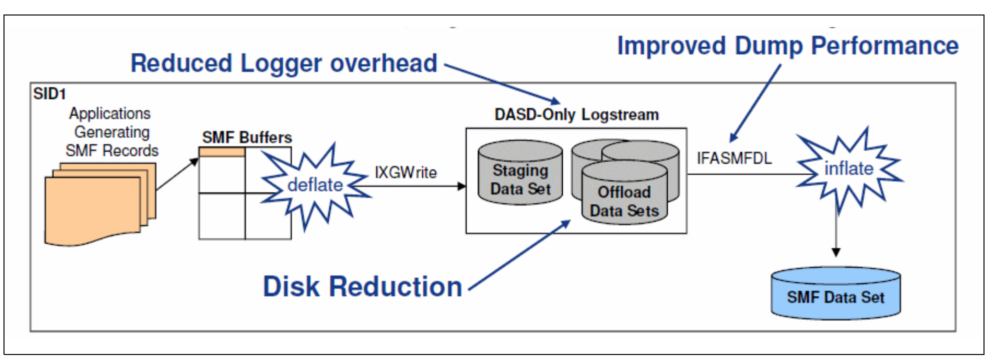


Figure 3-2 SMF use

All storage used for zEDC Express input/output (I/O) requires fixed storage for the input and output buffers. A new SMFPRMxx parameter, `PERMFIX`, is available as a subparameter of `COMPRESS` and as a global SMF parameter, enabling you to specify the amount of storage used for the SMF buffers that can remain permanently fixed. Each time SMF requests zEDC Express to compress a buffer, the buffers must be fixed. Doing this for each zEDC Express I/O operation increases the processing demands of the operation.

Setting the fixed storage to a high value can reduce the processing, but it might decrease the amount of fixed storage available to other applications.

3.3 Setting up z/OS SMF

The SMFPRMxx member in SYS1.PARMLIB controls the behavior of the SMF for your z/OS. To take advantage of zEDC with SMF data, you have to specify the new option `COMPRESS`.

The option can be added to either the `LSNAME` or `DEFAULTLSNAME` in the SMFPRMxx PARMLIB member.

If all zEDC Express features fail or none are available for use, message IFA730I is issued and SMF continues writing non-compressed records to the logstream. To restart a failed zEDC session, issue a `SETSMF RECORDING=LOGSTREAM` to try compression again, or alter other SMF parameters with the `SET` or the `SETSMF` command. Message IFA731I is issued when compression is successfully enabled.

Remember: You can have compressed and non-compressed log blocks in the same logstream:

- You can have one system writing compressed data to a shared logstream, and another system writing non-compressed data.

  If the zEDC feature is not available, SMF data is not compressed and the IFA730I message is issued.

- In the same system, you can switch back and forth between writing compressed and non-compressed log blocks. To switch from compressed back to non-compressed format, you `must` specify `NOCOMPRESS` on the `LSNAME` definition.

There is also another option, `PERMFIX(nnnM)` on the `COMPRESS` parameter, which enables the installation to dictate how much of the SMF buffer pool is permanently page fixed for zEDC; however, fixed pages are a constrained resource:

- A larger amount of storage improves the performance of SMF, but decreases the amount of storage available to other applications.
- A lower value for `PERMFIX` improves storage availability but can have a negative effect on zEDC performance.

`PERMFIX` can range from a minimum of 1 megabyte (MB) to a maximum of 2 gigabytes (GB). Due to processing needs, even if this value is `NOPERMFIX`, SMF can use up to 2 MB of fixed storage for zEDC usage.

If specified, this value overrides the global `PERMFIX` value.

New IFAQUERY and SMF type 23 record output fields can help the tailoring of the `PERMFIX` value.

Additional information about the SMFPRMxx PARMLIB member can be found in the chapter about SMFPRMxx (Product Ennoblement Policy) of the *z/OS MVS Initialization and Tuning Reference,* SA23-1380.

3.4 Dumping compressed records

When compressed data is processed by IFASMFDL (the SMF logstream dump program), it decompresses the SMF records for selection and writing.

SMF data is only compressed while it is resident in the System Logger. Ideally, IFASMFDL always runs on a z/OS V2R1 system with access to zEDC Express.

If the z/OS image does not meet this requirement, IFASMFDL returns an error with message IFA849I (Example 3-1).

Example 3-1 IFA849i message on SMF logstream dump program

```
IFA849I ENVIRONMENT ERROR. IFASMFDL FAILED DUE TO BAD SMF RECORD
                CONTENT IN LOGSTREAM IFASMF.#@$#PLEX.COMP. COMPRESSED
                DATA WAS UNEXPECTED. DIAGNOSTIC INFORMATION IFASMFDL
                Comp MBC 00000008 00000000
```

IFASEXIT, the logstream subsystem exit, which can be used by logstream owners to return records from a logstream to conventional data sets, cannot be used to read compressed SMF records.

The IFASMFDL program now has an option to force zEDC Express to be used when decompressing the data as it is retrieved from an SMF logstream. The alternative is to allow IFASMFDL to fall back to a software-based decompression, which would perform far worse than the zEDC Express.

If compressed SMF records must be read on a pre-z/OS V2R1 system, or on a system without access to zEDC Express, the new **SOFTINFLATE** parameter enables installations to process compressed SMF records using a software algorithm.

If you try to use IFASMFDL to read data from a zEDC-compressed SMF logstream on a system that does not have access to the zEDC Express feature, the job fails and generates a return code of 4 unless you specify **SOFTINFLATE** in the IFASMFDL control statements:

- If the zEDC feature is available, it is used.
- If the zEDC feature is *not* available, the CP is used to do the decompression (but this method can be central processing unit (CPU)-intensive).
- You could use the IBM z/OS Workload Manager (WLM) Scheduling Environments to direct SMF jobs to appropriate systems.

> **Important:** The ability to use **SOFTINFLATE** should be viewed as a fallback capability. All systems should have access to zEDC before enabling zEDC compression for SMF.

For additional information, see the chapter about Using the SMF Dump Programs in the *z/OS MVS System Management Facilities (SMF)* SA38-0667 manual.

3.5 Coexistence

Program temporary fixes (PTFs) for authorized program analysis report (APAR) OA41156 should be installed on z/OS V1R13 or V1R12 systems to tolerate the new SMFPRMxx keywords, and to enable the IFASMFDL SOFTINFLATE keyword and software decompression support.

On z/OS V2R1, any IFASMFDL job now needs to specify a region size of 4 MB or greater, because IFASMFDL has been enhanced to make the most of multi-block logstream browsing. This feature aids in processing compressed blocks and benefits all users.

If you decide to compress SMF records, you might want to consider the dumping environments that are used. You might also want to move or restrict IFASMFDL jobs to systems with access to zEDC Express for optimal performance.

Consider adding **SOFTINFLATE** to IFASMFDL jobs after investigating the cost, performance, and compatibility implications. Also note that entry-to-element ratios of CF structure-based logstreams might change as the data is compressed. The logger might encounter entry or element full conditions until it can resample to the new ratios.

3.6 Test case

To verify how well SMF works with zEDC, we perform the following steps:

- Environment setup
- SMF logstreams usage
- Comparison about IFASMFDL
  - Disk space
  - Elapsed time, CPU time, Service unit
- Comparison about IFASMFDP

3.6.1 Environment setup

Our test environment is a three-way sysplex, with all three systems writing all SMF data to a single shared logstream called `IFASMF.TYPDFLT`.

To test the SMF use of zEDC, you can use SMF to write the same SMF records to multiple logstreams. This enables you to get experience using zEDC without making any changes to existing SMF logstreams. Writing identical SMF records to both logstreams enables you to use System Logger type 88 SMF records to compare the amount of data being written to both logstreams.

To set up the environment, perform the following steps:

1. First, define two new logstreams with a retention period of two days (Example 3-2).

Example 3-2 Define two new logstreams

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
  DATA TYPE(LOGR) REPORT(YES)
```

```
DEFINE LOGSTREAM NAME(IFASMF.#@$#PLEX.NOCO)
       STRUCTNAME(IFASMF_TEST)
       LS_DATACLAS(LOGR24K) HLQ(IXGLOGR)
       MODEL(NO)
       LS_SIZE(100000) STG_DATACLAS(LOGR4K)
       STG_SIZE(150000) LOWOFFLOAD(0) HIGHOFFLOAD(60) STG_DUPLEX(NO)
       RETPD(2) AUTODELETE(YES) OFFLOADRECALL(YES)
       ZAI(NO) ZAIDATA('NO_ZAIDATA') WARNPRIMARY(NO)
       DASDONLY(NO) DIAG(NO) LOGGERDUPLEX(UNCOND)
       GROUP(PRODUCTION)
DEFINE LOGSTREAM NAME(IFASMF.#@$#PLEX.COMP)
       STRUCTNAME(IFASMF_TEST)
       LS_DATACLAS(LOGR24K) HLQ(IXGLOGR)
       MODEL(NO)
       LS_SIZE(100000) STG_DATACLAS(LOGR4K)
       STG_SIZE(150000) LOWOFFLOAD(0) HIGHOFFLOAD(60) STG_DUPLEX(NO)
       RETPD(2) AUTODELETE(YES) OFFLOADRECALL(YES)
       ZAI(NO) ZAIDATA('NO_ZAIDATA') WARNPRIMARY(NO)
       DASDONLY(NO) DIAG(NO) LOGGERDUPLEX(UNCOND)
       GROUP(PRODUCTION)
```

2. For our testing with zEDC, we set up SMF with two new logstreams, one compressed, one not compressed, and used SMF to send the same record type to both logstreams.

   The initial IFASMFxx parmlib member includes the statement shown in Example 3-3.

   Example 3-3 Starting IFASMFxx member

```
RECORDING(LOGSTREAM)
DEFAULTLSNAME(IFASMF.TYPDFLT)
```

3. We add two new logstreams, `IFASMF.#@$#PLEX.COMP` to collect all SMF records in compress mode and `IFASMF.#@$#PLEX.NOCO` for noncompress mode (Example 3-4).

   Example 3-4 Updated IFASMFxx parmlib member

```
RECORDING(LOGSTREAM)
LSNAME(IFASMF.#@$#PLEX.COMP,TYPE(0:255),COMPRESS)
LSNAME(IFASMF.#@$#PLEX.NOCO,TYPE(0:255))
DEFAULTLSNAME(IFASMF.TYPDFLT,COMPRESS)
```

   We use the same SMFPRMxx for systems that have zEDC and those that do not.

   The systems that don't have zEDC get an IFA730E error message, and log blocks written from that system will not be compressed, but the logstream will be used. When that system is migrated to z/OS 2.1 and to a logical partition (LPAR) that is connected to zEDC, it starts compressing its SMF log blocks with no further changes required.

4. We activate the same member `SMFPRMZA` on the **`#@$A`** system that has zEDC (Example 3-5).

   Example 3-5 SET SMF=ZA command on #@$A with zEDC

```
SET SMF=ZA
IEE252I MEMBER SMFPRMZA FOUND IN SYS1.PARMLIB
IEE967I 09.53.57 SMF PARAMETERS
        MEMBER = SMFPRMZA
.......
        SID(#@$A) -- DEFAULT
```

```
.......
       DEFAULTLSNAME(IFASMF.TYPDFLT) -- PARMLIB
       LSNAME(IFASMF.#@$#PLEX.NOCO,TYPE(0:255)) -- PARMLIB
       LSNAME(IFASMF.#@$#PLEX.COMP,COMPRESS,TYPE(0:255)) -- PARMLIB
       RECORDING(LOGSTREAM) -- PARMLIB

.......
       ACTIVE -- PARMLIB
IFA716I THERE ARE NO RECORDS FOR DEFAULT LOGSTREAM TO COLLECT
DEFAULTLSNAME(IFASMF.TYPDFLT) PARAMETER IS IGNORED.
.......
IFA711I LOGSTREAM PARAMETERS ARE IN EFFECT
IFA714I 09.54.03 SMF STATUS
         LOGSTREAM NAME               BUFFERS       STATUS
       A-IFASMF.#@$#PLEX.COMP               0       CONNECTED
       A-IFASMF.#@$#PLEX.NOCO               0       CONNECTED
IEE536I SMF      VALUE ZA NOW IN EFFECT
IFA731I COMPRESSION ACTIVE FOR SMF
 FOR IFASMF.#@$#PLEX.COMP
```

5. On the **#@$2** system that does not have zEDC, the activation results in the messages shown in Example 3-6.

Example 3-6 SET SMF=ZA command on #@$2 without zEDC

```
SET SMF=ZA
IEE252I MEMBER SMFPRMZA FOUND IN SYS1.PARMLIB
IEE967I 11.14.01 SMF PARAMETERS
       MEMBER = SMFPRMZA
.......
       SID(#@$2) -- DEFAULT
.......
       DEFAULTLSNAME(IFASMF.TYPDFLT,COMPRESS) -- PARMLIB
       LSNAME(IFASMF.#@$#PLEX.NOCO,TYPE(0:255)) -- PARMLIB
       LSNAME(IFASMF.#@$#PLEX.COMP,COMPRESS,TYPE(0:255)) -- PARMLIB
       RECORDING(LOGSTREAM) -- PARMLIB
.......
       ACTIVE -- PARMLIB
IFA716I THERE ARE NO RECORDS FOR DEFAULT LOGSTREAM TO COLLECT
DEFAULTLSNAME(IFASMF.TYPDFLT) PARAMETER IS IGNORED.
.......
IXL014I IXLCONN REQUEST FOR STRUCTURE IFASMF_TEST
WAS SUCCESSFUL.  JOBNAME: IXGLOGR ASID: 001B
CONNECTOR NAME: IXGLOGR_#@$2 CFNAME: FACIL06
IFA711I LOGSTREAM PARAMETERS ARE IN EFFECT
IFA714I 11.14.06 SMF STATUS
         LOGSTREAM NAME               BUFFERS       STATUS
       A-IFASMF.#@$#PLEX.COMP               0       CONNECTED
       A-IFASMF.#@$#PLEX.NOCO               0       CONNECTED
IEE536I SMF      VALUE ZA NOW IN EFFECT
IFA730E COMPRESSION FAILED FOR SMF
 FOR IFASMF.#@$#PLEX.COMP
 DIAGNOSTIC INFORMATION IFALS834 Register 00000004 00000000
```

3.6.2 SMF logstreams usage

After the environment setup, the SMF recording on our three systems is as shown in Example 3-7. These numbers are only related to in-memory buffers, and have no relation to how much data is in a logstream, or to compression.

Example 3-7 The three systems SMF recording

```
RO *ALL,D SMF
IEE421I RO *ALL,D SMF 968
#@$A     RESPONSES ---------------------------------------------------
IFA714I 11.17.46 SMF STATUS 967
          LOGSTREAM NAME                BUFFERS        STATUS
        A-IFASMF.#@$#PLEX.COMP            151364       CONNECTED
        A-IFASMF.#@$#PLEX.NOCO             20784       CONNECTED
#@$2     RESPONSES ---------------------------------------------------
IFA714I 11.17.46 SMF STATUS 757
          LOGSTREAM NAME                BUFFERS        STATUS
        A-IFASMF.TYPDFLT                   39466       CONNECTED
#@$3     RESPONSES ---------------------------------------------------
IFA714I 11.17.46 SMF STATUS 310
          LOGSTREAM NAME                BUFFERS        STATUS
        A-IFASMF.TYPDFLT                   47443       CONNECTED
```

We focus the evaluation on **#@$A** system, it is the only one to use the two new logstreams.

In a day, the **#@$A** system writes the amount of SMF records listed in Example 3-8.

Example 3-8 One day SMF records amount

```
LSNAME                        START DATE/TIME       END DATE/TIME
IFASMF.#@$#PLEX.COMP       11/10/2014 00:00:18 11/10/2014 23:59:49

                                        SUMMARY ACTIVITY REPORT
    START DATE-TIME  11/09/2014-23:59:35                       END DATE-TIME  11/11/2014-11:30:08
    RECORD       RECORDS         PERCENT      AVG. RECORD   MIN. RECORD   MAX. RECORD       RECORDS
      TYPE          READ        OF TOTAL           LENGTH        LENGTH        LENGTH       WRITTEN
         2             0                                                                          1
         3             0                                                                          1
        14           409           .02 %           458.60           416           704           273
        15           455           .03 %           421.78           416           525           335
        17            67           .00 %           100.00           100           100             0
        22           424           .02 %            60.00            60            60             0
        23           213           .01 %         4,170.00         4,170         4,170             0
        26           324           .02 %           514.58           514           522             0
        30        13,867           .78 %         1,714.16           480        32,070         9,313
        32            47           .00 %           250.80           224           308             0
        41           170           .01 %           368.18           146           412             0
        42         7,905           .44 %         1,818.22           176        22,960         5,316
        60        26,197          1.47 %           636.90           342         1,279             0
        61           106           .01 %           332.48           301           411             0
        62            68           .00 %           188.00           188           188             0
        64        26,027          1.46 %           474.00           474           474             0
        65           129           .01 %           351.68           282           407             0
        66            32           .00 %           309.25           304           325             0
        70           426           .02 %        14,658.27         1,196        32,156           285
        71           142           .01 %         2,036.00         2,036         2,036            95
        72        10,366           .58 %         1,528.16         1,132        13,260         6,935
        73           142           .01 %        21,815.74        21,780        21,816            95
        74        17,660           .99 %        19,976.88           364        32,726        11,780
        75           569           .03 %           264.00           264           264           380
        77           142           .01 %           414.64           160           960            95
        78           426           .02 %        14,169.92         1,888        32,416           285
        88         7,846           .44 %           256.83           161           308             0
        89           321           .02 %           908.55           426         1,082             0
        90             2           .00 %         1,064.00         1,064         1,064             0
        91           142           .01 %           311.00           311           311             0
        92     1,529,839         86.03 %           187.76           160           348             0
       100        21,360          1.20 %         1,487.33           146         5,310             0
       102        13,347           .75 %         1,216.30           226         3,102             0
```

```
    110        96,764          5.44 %        18,215.32           186        32,742             0
    113         2,264           .13 %           781.00           778           784             0
                                      SUMMARY ACTIVITY REPORT
 START DATE-TIME  11/09/2014-23:59:35                               END DATE-TIME  11/11/2014-11:30:08
 RECORD       RECORDS         PERCENT      AVG. RECORD   MIN. RECORD   MAX. RECORD       RECORDS
  TYPE           READ        OF TOTAL           LENGTH        LENGTH        LENGTH       WRITTEN
 TOTAL      1,778,198           100 %         1,437.00            60        32,742        35,189
 NUMBER OF RECORDS IN ERROR              0
```

The disk space usage for logstreams DASD log data sets is listed in Example 3-9.

Example 3-9 Logstreams DASD log data sets tracks

```
DSLIST - Data Sets Matching IXGLOGR.IFASMF.#@$#PLEX.**.DATA       Row 1 of 12
Command ===>                                                  Scroll ===> CSR

Command - Enter "/" to select action                        Tracks %Used   XT
-------------------------------------------------------------------------------
         IXGLOGR.IFASMF.#@$#PLEX.COMP.A0000002.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.COMP.A0000003.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000022.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000023.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000024.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000025.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000026.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000027.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000028.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000029.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000030.DATA           8340    ?     1
         IXGLOGR.IFASMF.#@$#PLEX.NOCO.A0000031.DATA           8340    ?     1
***************************** End of Data Set list ****************************
```

The ratio is ten to two. It is important to remember that a logstream DASD log data set is deleted when all data is expired. The IXGRPT2 report (Example 3-11 on page 41) better evidences the volume of data written on logstream.

The LOGR details about the two logstreams are shown in Example 3-10.

Example 3-10 IXCMIAPU LIST with DETAIL(YES)

```
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
  DATA TYPE(LOGR) REPORT(NO)
  LIST LOGSTREAM NAME(IFASMF.#@$#PLEX.NOCO) DETAIL(YES)
  LIST LOGSTREAM NAME(IFASMF.#@$#PLEX.COMP) DETAIL(YES)

DATA SET NAMES IN USE: IXGLOGR.IFASMF.#@$#PLEX.NOCO.<SEQ#>

    Ext.   <SEQ#>    Lowest Blockid /  Highest GMT /      Highest Local /    Status
                     Highest Blockid   Highest RBA        System Name
    -----  --------  ----------------  -----------------  -----------------  -------------------------
   *00001  A0000022  00000002197E1123  11/09/14 21:50:00  11/09/14 16:50:00
                     0000000231EB8B89  186E7A56           #@$A
           A0000023  0000000231EC8B79  11/10/14 03:20:00  11/09/14 22:20:00
                     000000024A599B47  186E0FBE           #@$A
           A0000024  000000024A5A9B37  11/10/14 08:54:23  11/10/14 03:54:23
                     0000000262C86DC7  186ECFAB           #@$A
           A0000025  0000000262C96AE2  11/10/14 14:29:11  11/10/14 09:29:11
                     000000027B372FE6  186EC4C0           #@$A
           A0000026  000000027B382FA2  11/10/14 19:44:35  11/10/14 14:44:35
                     0000000293A57D25  186E185F           #@$A
           A0000027  0000000293A64801  11/11/14 01:20:00  11/10/14 20:20:00
```

```
                00000002AC134C87  186E046E           #@$A
      A0000028  00000002AC144C6F  11/11/14 06:39:30  11/11/14 01:39:30
                00000002C4824B87  186EFE70           #@$A
      A0000029  00000002C4834ADF  11/11/14 12:00:00  11/11/14 07:00:00
                00000002DCF071B0  186E0C0F           #@$A
      A0000030  00000002DCF156EE  11/11/14 17:20:00  11/11/14 12:20:00
                00000002F55FADEA  186ED756           #@$A
      A0000031  00000002F5602E44  11/11/14 18:14:35  11/11/14 13:14:35  CURRENT
                00000002F932CCC4  03D37674           #@$A

NUMBER OF DATA SETS IN LOGSTREAM: 10


DATA SET NAMES IN USE: IXGLOGR.IFASMF.#@$#PLEX.COMP.<SEQ#>

Ext.  <SEQ#>    Lowest Blockid /  Highest GMT /      Highest Local /    Status
                Highest Blockid   Highest RBA        System Name
----- --------  ----------------  -----------------  -----------------  -------------------------
*00001 A0000002  0000000030DDF7A6  11/11/14 07:36:03  11/11/14 02:36:03
                00000000494CD293  186EE9B7           #@$A
      A0000003  00000000494CE15D  11/11/14 13:57:54  11/11/14 08:57:54  CURRENT
                000000004C2F1D27  02E26D98           #@$A

NUMBER OF DATA SETS IN LOGSTREAM: 2
```

We use the IXGRPT2 sample, available in SYS1.SAMPLIB, to investigate the System Logger Type 88 SMF records to compare the amount of data being written to both logstreams (Example 3-11).The ratio is 10 to one.

Example 3-11 IXGRPT2 to compare amount of data written to logstreams

```
LOGSTREAM  IFASMF.#@$#PLEX.COMP

     TME        DTE   SYN          LWI              LIB              LAB              LWB              LDB
 --------  ---------- --------  ---------  ----------------  ----------------  ----------------  ----------------
 00:14:35  2014/11/11  #@$A          384              1618             15848           2253001                0
 ........  2014/11/11  #@$A          272              1618             15864           1735760                0
 14:29:35  2014/11/11  #@$A          224              1611             16963           1354439                0

 AVERAGE                             314              1614             16285           1928252          2270305


 LOGSTREAM  IFASMF.#@$#PLEX.NOCO

     TME        DTE   SYN          LWI              LIB              LAB              LWB              LDB
 --------  ---------- --------  ---------  ----------------  ----------------  ----------------  ----------------
 00:14:35  2014/11/11  #@$A          397             32734             65532          23842148                0
 ........  2014/11/11  #@$A          257             32734             65532          15664262                0
 14:29:35  2014/11/11  #@$A          238             32734             65532          14539670                0

 AVERAGE                             314             32734             65532          19020579         19197165

LWI Number of IXGWRITES
LIB Min Blocklen used
LAB Max blocklen used
LWB Bytes written total
LDB Bytes written to DASD
```

The SMF Type 23 (SMF statistics) records have also been updated to add new fields about zEDC:

- SMF23LFG contains flags to indicate if zEDC is being used by this logstream.
- SMF23CWN contains the number of compressed log blocks written to the logstream.
- SMF23NCN contains the number of uncompressed log blocks written to the logstream.

3.6.3 Comparison about IFASMFDL

We run the SMF logstream dump program, IFASMFDL, to dump all records from compressed SMF logstream on the system that had access to the zEDC Express feature, and on the system that did not.

We specify the **SOFTINFLATE** parameter on the IFASMFDL utility, to verify that the software decompression is used when hardware is unavailable.

The number of records handled is shown in Example 3-12.

Example 3-12 Summary report IFASMFDL

```
SUMMARY ACTIVITY REPORT
START DATE-TIME  11/11/2014-02:31:28                        END DATE-TIME  11/13/2014-13:52:16
RECORD       RECORDS          PERCENT      AVG. RECORD   MIN. RECORD   MAX. RECORD       RECORDS
  TYPE          READ         OF TOTAL           LENGTH        LENGTH        LENGTH       WRITTEN
TOTAL      2,976,437            100 %         1,427.90            60        32,742     2,976,439
NUMBER OF RECORDS IN ERROR              0
```

Results are shown in Table 3-1.

Table 3-1 SMF differences with and without zEDC

LPAR	Output data set	Tracks	EXCPs	CPU time	Elapsed time	Service units
With zEDC	Compressed	7,665	7,699	0.03	0.16	757,000
With zEDC	Not compressed	77,505	155,000	0.02	0.71	396,000
Without zEDC	Not compressed (input logstream compressed)	77,505	155,000	1.95	2.63	34,919,000

The columns in the table contain the following information:

LPAR We use the same IBM zEnterprise EC12 (zEC12).

Output data set We use a specific DataClass to compress output.

Tracks Disk space.

EXCP I/O count.

CPU time Processing time used to run the program in minutes.

Elapsed time Job length in minutes.

Service units Metric used by z/OS to measure the CPU consumption by transactions running under z/OS processes.

3.6.4 Comparison about IFASMFDP

We use the SMF data set dump program, IFASMFDP, as a test case in 5.3, “Example of zBNA zEDC Express analysis” on page 80.

SMF data is a good candidate to be archived on a sequential data set using zEDC.

3.7 zEDC and PCIe monitoring

An IBM Resource Management Facility (RMF) Postprocessor `PCIe Activity Report` is available in Extensible Markup Language (XML) output format. The report provides measurements about the activity of PCIe-based functions and their use of hardware accelerators. A PCIe function is captured by the report if one of the following hardware feature activities has been measured:

- Remote Direct Memory Access (RDMA) over Converged Enhanced Ethernet
- zEDC capability using zEDC Express

In addition, RMF provides new overview conditions for the Postprocessor based on a new subtype 9 of SMF record 74.

You can obtain such reports in two ways:

- Using RMF Postprocessor batch job:
  - Install Postprocessor XML toolkit.
  - Run RMF Postprocessor batch job.
  - View XML reports.
- Using RMF Spreadsheet Reporter

3.7.1 Using RMF Postprocessor batch job

In this section, we examine the RMF support for zEDC.

Install Postprocessor XML toolkit

The Postprocessor XML Toolkit is part of the RMF product. The application files and RMF installation utility of the Postprocessor XML Toolkit are provided in the ERBXMLTK member of the SERBPWSV host distribution library. To install the toolkit, complete the following steps:

1. Download the ERBXMLTK member as the binary file `erbxmltk.msi` (Figure 3-3).

Figure 3-3 Download XML Toolkit

2. Double-click the `.msi` package file to install the `.msi` package using the Windows Installer.

3. Pick out the XML toolkit directory (Figure 3-4).

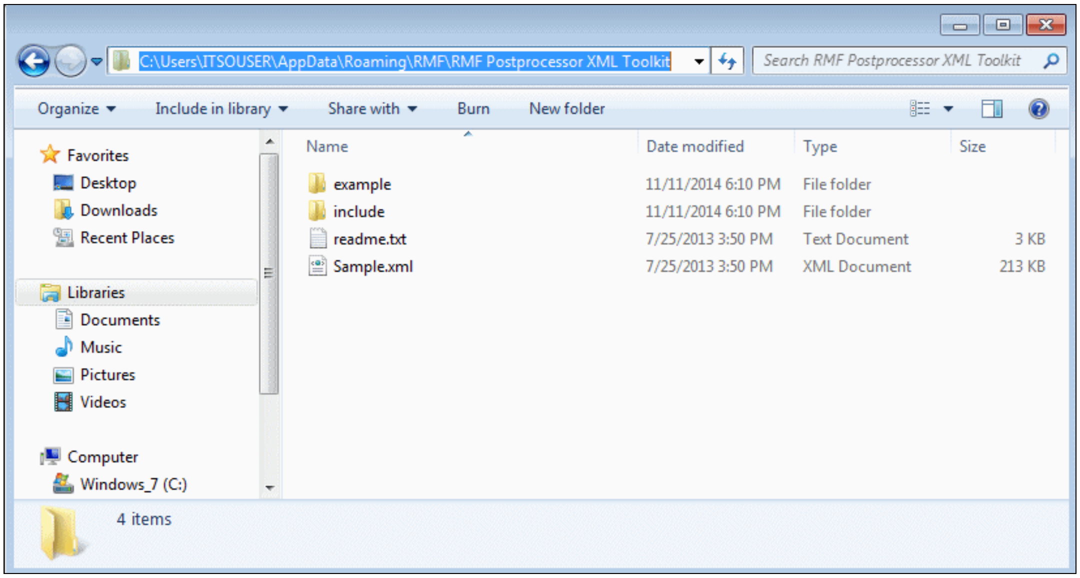


Figure 3-4 XML Toolkit directory

Run the RMF Postprocessor batch job

We use the following job to create a PCIe report in XML format. Example 3-13 shows the ddname **XPRPTS** to address the output and the required **REPORTS(PCIE)** SYSIN parameter.

Example 3-13 RMF Postprocessor sample

```
//EXTR      EXEC PGM=IFASMFDL,REGION=20M
//SYSPRINT DD SYSOUT=*
//SMFOUT    DD DSN=PBRES3.NOZEDC.SMF1,DISP=(,CATLG),UNIT=SYSDA,
//          SPACE=(CYL,(500,50),RLSE),RECFM=VBS,LRECL=32760
//SYSIN DD *
 LSNAME(IFASMF.#@$#PLEX.COMP,OPTIONS(DUMP))
 SOFTINFLATE
 OUTDD(SMFOUT,TYPE(70:79))
 DATE(2014315,2014315)
 START(0800)
 END(1200)
//RMFSORT   EXEC PGM=SORT,REGION=0M
//SORTIN    DD    DISP=SHR,DSN=PBRES3.NOZEDC.SMF1
//SORTOUT   DD    DISP=(NEW,PASS),UNIT=SYSDA,SPACE=(TRK,(2000,900)),
//          DSN=PBRES3.NOZEDC.SMF1.RMF
//SORTWK01 DD    DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(1000,500))
//SORTWK02 DD    DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(1000,500))
//SORTWK03 DD    DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(TRK,(1000,500))
//SYSPRINT DD    SYSOUT=*
//SYSOUT    DD    SYSOUT=*
//SYSIN     DD    *
  SORT FIELDS=(11,4,CH,A,7,4,CH,A),EQUALS
  MODS E15=(ERBPPE15,500,,N),E35=(ERBPPE35,500,,N)
//RMFPP     EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD    DISP=(OLD,DELETE),DSN=*.RMFSORT.SORTOUT
//MFPMSGDS DD    SYSOUT=*
//XPRPTS    DD    DISP=(,CATLG),DSN=PBRES3.RMF.XPRPTS.XML,
//          UNIT=SYSDA,RECFM=VB,LRECL=8192,BLKSIZE=0,
```

```
//          SPACE=(TRK,(1000,100),RLSE)
//SYSIN     DD    *
  DATE(11112014,11112014)
  SUMMARY(INT,TOT)
  REPORTS(PCIE)
/*
```

View XML reports

To view the XML reports, complete the following steps:

1. Download the `PBRES3.RMF.XPRPTS.XML` XML output data set into the `C:\Users\ITSOUSER\AppData\Roaming\RMF\RMF Postprocessor XML Toolkit` Postprocessor XML Toolkit directory on your workstation using a `.xml` file extension.

   > **Tip:** Download the data set containing the XML output of the Postprocessor reports in ASCII format to the Postprocessor XML Toolkit directory.

2. Open the XML Postprocessor reports within the Postprocessor XML Toolkit using a browser. The PCIe Activity Report looks like that shown in Figure 3-5.

   The meanings of the fields are described in 3.7.3, “Fields in the RMF PCIe report” on page 49.

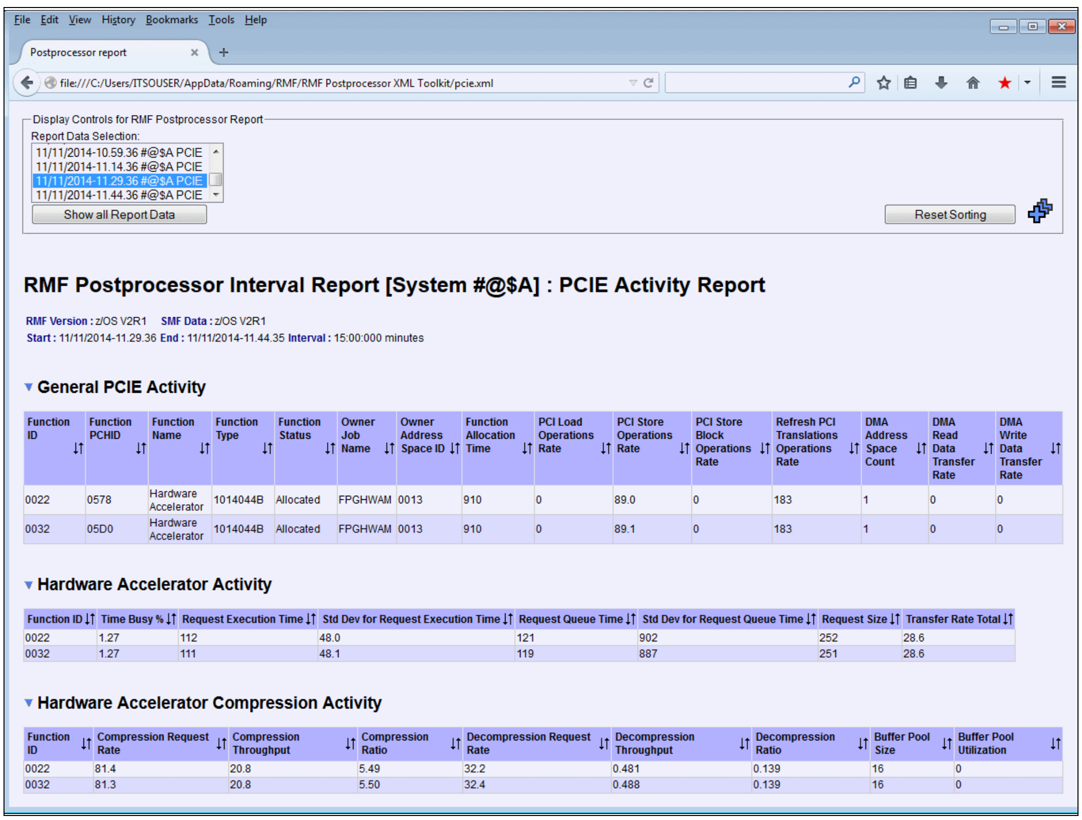


Figure 3-5 XML Toolkit view report

3.7.2 RMF Spreadsheet Reporter

The `IBM RMF Spreadsheet Reporter Java TM Technology Edition` provides built-in support for the new Postprocessor XML-formatted reports. You can request the new XML format by using the general option **Use XML Report Format**:

1. Modify Options in **Settings** → **Options**. Figure 3-6 shows this sequence.

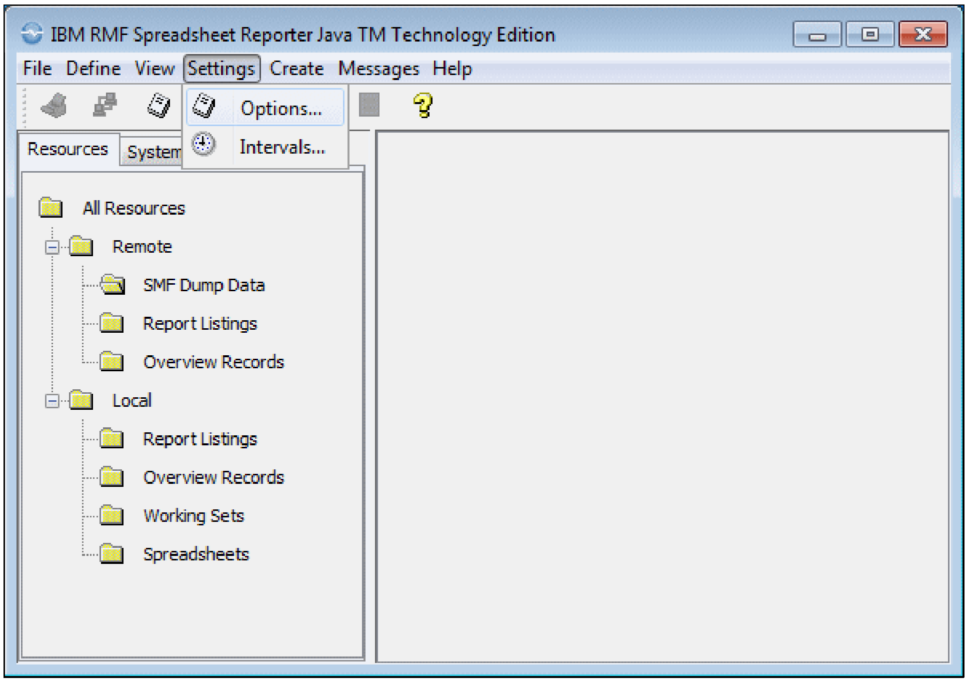


Figure 3-6 IBM RMF Spreadsheet Reporter Settings

2. The `Options` window opens. We use this window to select options, as shown in Figure 3-7:

 a. On the `General` tab, select **Use XML Report Format**.

 b. On the `Reports` tab, select **PCIe Activity**.

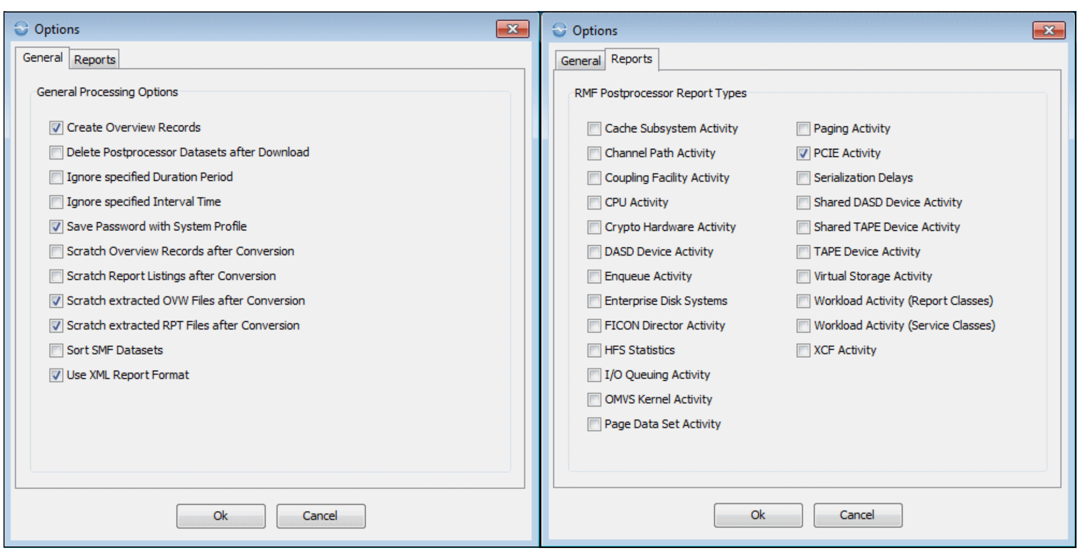


Figure 3-7 IBM RMF SR Options: General and Reports

3. To create a Report Listing, on the `navigation pane` (left side), we open resource type SMF Dump Data (Figure 3-8) to select one or more remote SMF data sets as input to the `Create Report Listing` dialog.

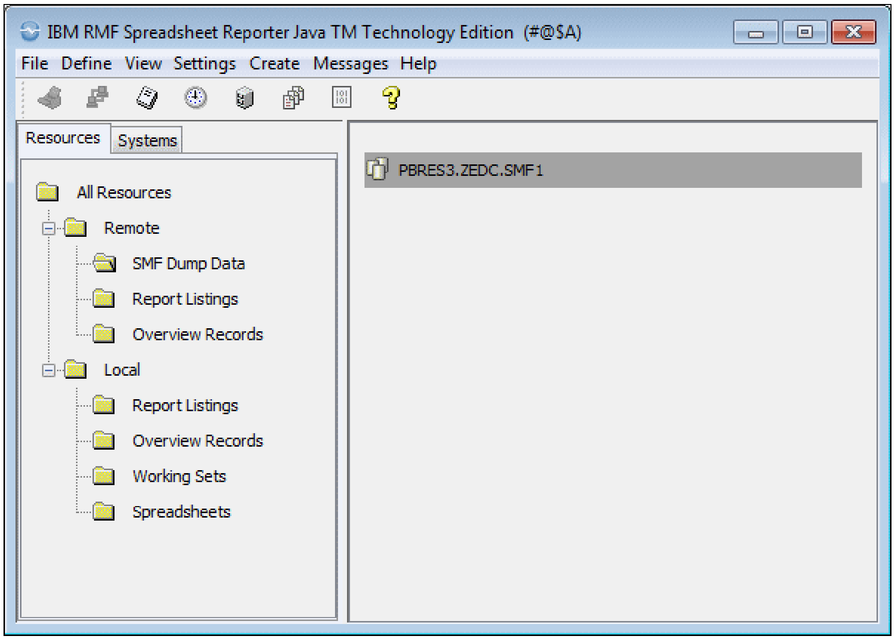


Figure 3-8 RMF SR Create Report Listing_1

4. Now after opening the Create menu, you see that the `Report Listing` item is enabled (Figure 3-9).

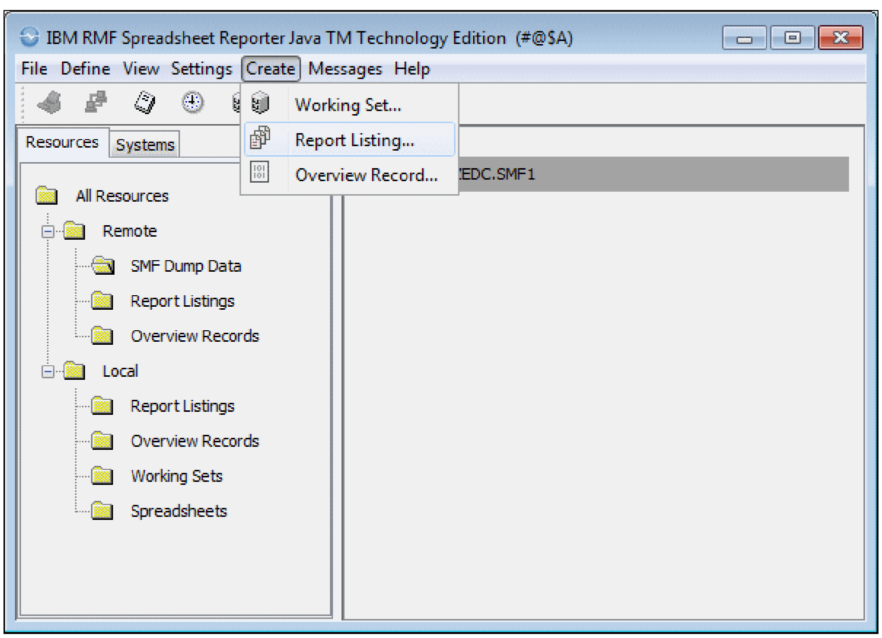


Figure 3-9 RMF SR Create Report Listing_2

5. Clicking `Report Listing`. This item opens the `Create Report Listing` dialog. With this dialog, you can generate a Postprocessor job and start it on the remote system.

6. Indicate `PCIE_ZEDC_Report.xml` as the local name for the report (Figure 3-10).

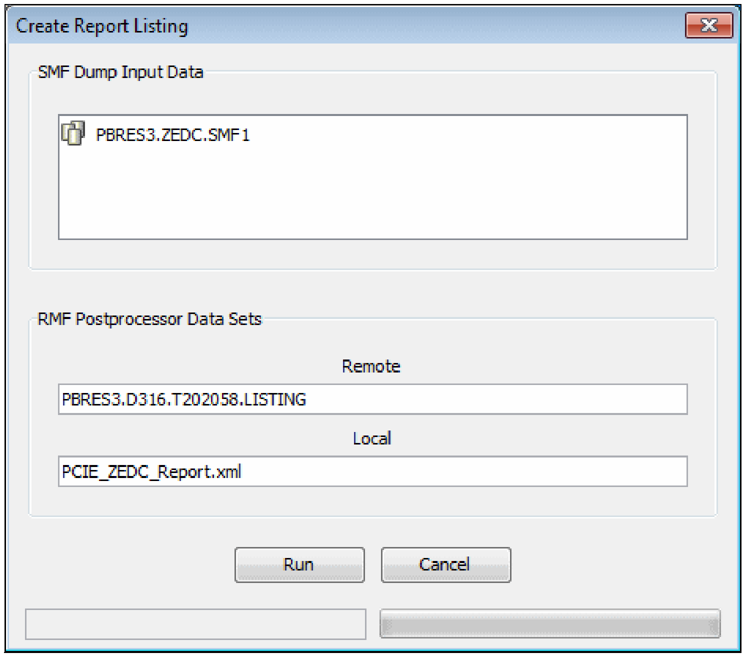


Figure 3-10 RMF SR Create Report Listing_3

7. There are two ways to view local Report Listings. On the `navigation pane` (left side), open Local type Record Listing, then perform one of the following actions:
   – Double-click a local Report Listing in the `view pane` (right side).
   – Select an entry in the `view pane` (right side), click the right mouse button and then select View from the menu (Figure 3-11).

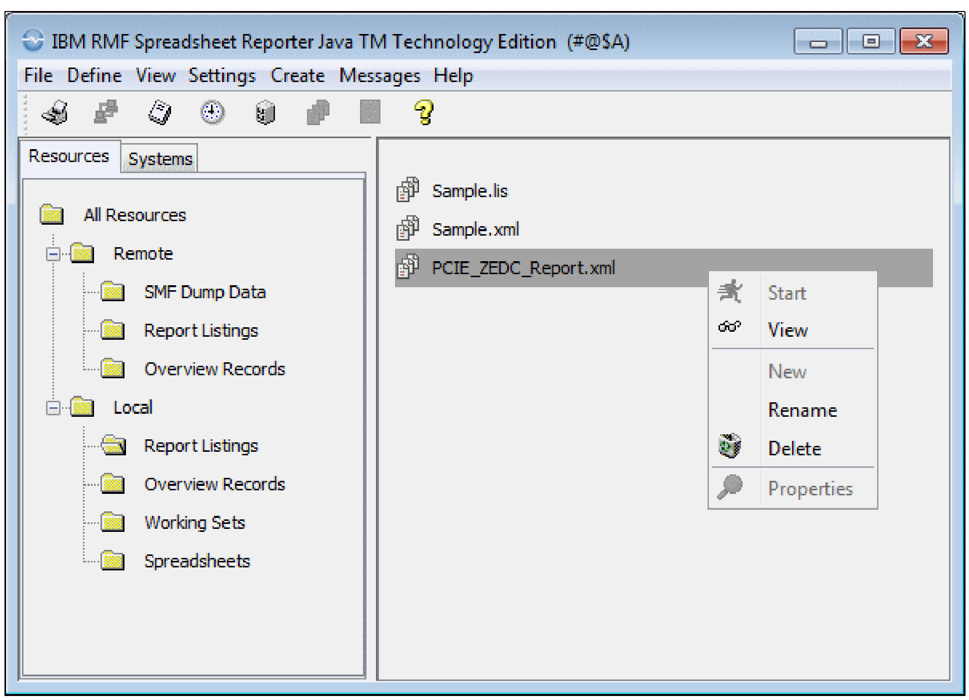


Figure 3-11 RMF Spreadsheet Reporting Report Listing view

8. Report Listings with extension `.xml` are opened in a web browser.

3.7.3 Fields in the RMF PCIe report

The PCIe Activity Report is divided into three sections: General PCIe Activity, Hardware Accelerator Activity, and Hardware Accelerator Compression Activity. The following list describes these sections:

- General PCIe Activity

  The General PCIe Activity section shows measurements for all PCIe functions that are independent from the type of the used hardware feature. The measurements reflect the activity of the z/OS system on which RMF data collection took place. They consist of data rates about the communication of z/OS programs with PCIe functions by using PCI operations that are transferring data blocks from z/OS to the PCIe function (**PCI LOAD**, **PCI STORE**, **PCI STORE BLOCK**, and **REFRESH PCI TRANSLATIONS**).

  They also consist of measurements of data transfers from the PCIe function to direct memory access (DMA) address spaces that are in z/OS main storage (DMA read/write counters).

- Hardware Accelerator Activity and Hardware Accelerator Compression Activity

  The Hardware Accelerator Activity section and the Hardware Accelerator Compression Activity section have single-system scope, and are using the measurements displayed in the General PCIe Activity section. They are only displayed if the zEDC hardware feature is used for compression acceleration. In this case, they display the following information:

  - Common accelerator metrics, such as total request execution time, or the amount of transferred data
  - Compression-specific metrics, such as the amount of compressed data and the number and throughput of compression requests
  - Device driver buffer statistics

Figure 3-12 shows an example of such report and the following tables include the meaning of the fields for:

– Hardware Accelerator Activity (Table 3-2)
– Hardware Accelerator Compression Activity (Table 3-3 on page 51)

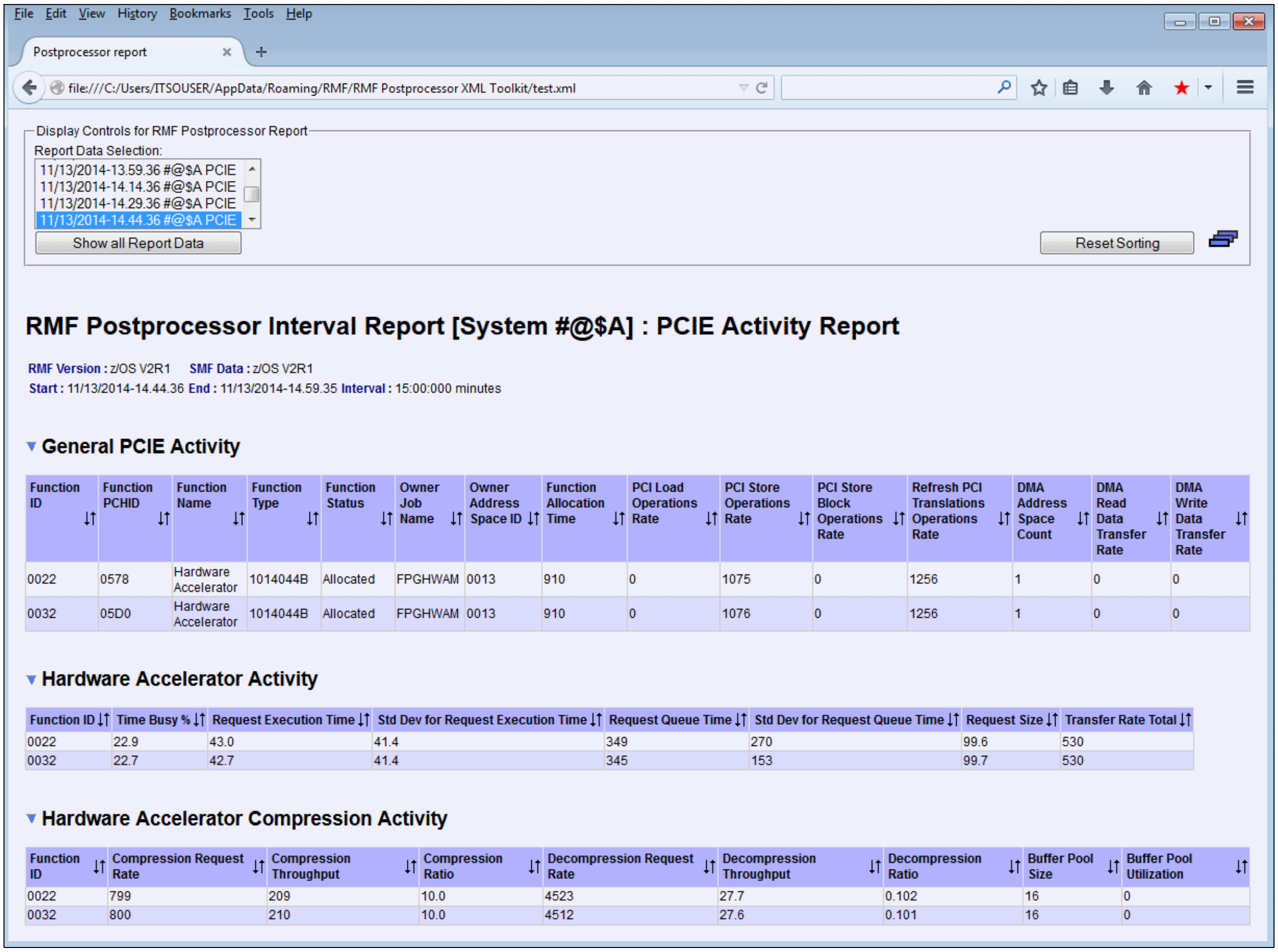


Figure 3-12 RMF XML PCIe Activity Report

Table 3-2 Hardware Accelerator Activity fields

Field Heading	**Our example**	**Meaning**
Time Busy %	**22.9**	Percentage of time adapter was busy by this system.
Request Execution Time	**43.0**	Average time in microseconds (µs) to process a request from this z/OS.
Request Queue Time	**349**	Time in µs blocks were waiting to be sent to zEDC. Consider that the exploiter might queue several blocks before sending to zEDC.
Request Size	**99.6**	Average sum size in KB of blocks sent to and from zEDC.
Transfer Rate Total	**530.0**	Number of MB per second transferred by DMA operations

Table 3-3 Hardware Accelerator Compression Activity fields

Field Heading	**Our example**	**Meaning**
Compression Request Rate	**799**	Number of compression requests per second
Compression Throughput	**209**	MB compressed per second
Compression Ratio	**10**	Average compression ratio for this LPAR
Decompression Request Rate	**4523**	Number of decompression requests per second
Decompression Throughput	**27.7**	MB of the compressed data decompressed per second
Decompression Ratio	**0.102**	Average decompression ratio for this LPAR
Buffer Pool Size	**16**	Total size of memory in MB allocated to the buffer pool

The fields are described in the IBM Resource Measurement Facility™ paper, *IBM Resource Measurement Facility Report Analysis,* SC34-2665.

4

z/OS zEnterprise Data Compression Express feature and BSAM/QSAM data sets

This chapter describes the handling of compressed sequential files allocated by using basic sequential access method (BSAM) or queued sequential access method (QSAM) with the IBM zEnterprise Data Compression Express (zEDC Express) feature enabled. This chapter provides a short overview of the BSAM and the QSAM. You find the necessary steps needed to modify the IGDSMSxx PARMLIB member, and a description of the parameters.

This chapter then describes the changes to Data Facility Storage Management Subsystem (DFSMS) as a prerequisite to the employment of the zEDC Express feature. We then describe how to allocate BSAM and QSAM files that are eligible for compression by the zEDC feature.

This chapter contains the following sections:

- BSAM and QSAM
- System setup and DFSMS parameters
- Work with zEDC compressed files
- Identifying candidates
- IBM DB2 for z/OS data and zEDC

© Copyright IBM Corp. 2015. All rights reserved.

4.1 BSAM and QSAM

Both BSAM and QSAM support the definition of sequential data sets.

4.1.1 Basic Sequential Access Method

BSAM arranges records sequentially in the order in which they are entered. A data set that has this organization is a sequential data set. It enables programs to read and write physical blocks of data. The user organizes records with other records into blocks. This is basic access. You can use BSAM with the following data types:

- Basic format sequential data sets (before z/OS V1.7, these were known as *sequential data sets* or more accurately as *non-extended-format sequential data sets*)
- Large format sequential data sets
- Extended-format data sets
- z/OS UNIX files

Figure 4-1 depicts the user managing the block to retrieve records.

Figure 4-1 Records and blocks with BSAM

4.1.2 Queued Sequential Access Method

QSAM arranges records sequentially in the order that they are entered to form sequential data sets, which are the same as those data sets that BSAM creates. The system organizes records with other records, and it enables programs to access logical records within physical blocks of data. QSAM anticipates the need for records based on their order. To improve performance, QSAM reads these records into storage before they are requested. This is called *queued access*. You can use QSAM with the following data types:

- Basic format sequential data sets (before z/OS V1.7, these were known as *sequential data sets* or more accurately as *non-extended-format sequential data sets*)
- Large format sequential data sets
- Extended-format data sets
- z/OS UNIX files

Figure 4-2 depicts the user accessing the records.

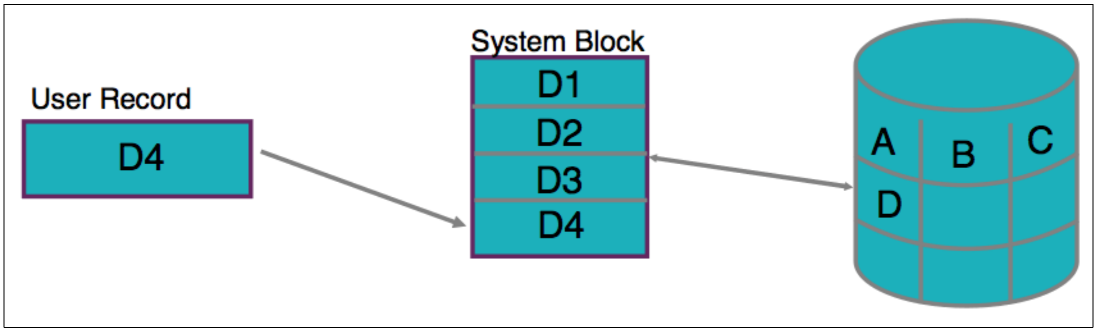


Figure 4-2   Records and blocks with QSAM

4.2 System setup and DFSMS parameters

zEDC Express (FC#0420) is a hardware feature card that fits into the PCIe input/output (I/O) drawer. To enable the zEDC Express feature on the system, the following prerequisites have to be completed:

1. Order and physically install the zEDC Express feature card(s) (FC#0420) into the machine. The zEDC Express feature is exclusive to IBM zEnterprise EC12 (zEC12), IBM zEnterprise BC12 (zBC12), and later machines.

   > **Note:** Minimum microcode library (MCL) requirement is the March 31, 2014 Firmware MCL release for zEC12 and zBC12. See also 2.2, "z/OS: Verify the prerequisites" on page 12.
   >
   > Each feature can be shared across up to 15 LPARs; up to 8 features available on zEC12, zBC12 or newer machines.

2. The zEDC Express for z/OS priced software feature must be enabled. Use the new IFAPRDxx member in SYS1.PARMLIB. See 2.3, "z/OS: Enabling the Priced Software Feature" on page 14.
3. Define the zEDC Express feature in the hardware configuration definition (HCD) and make it available to the system. See 2.5, "HCD: Defining the device" on page 17. Schedule an initial program load (IPL) for each logical partition (LPAR) on which the zEDC Express feature should become active.

   > **Note:** An IPL is mandatory for new or changed content of IFAPRDxx member to be recognized by z/OS.

   This concludes the physical part of the installation and activation of the zEDC Express feature.

The zEDC Express feature also needs to be activated on the DFSMS level. zEDC compression for new data sets can be requested in a similar manner to how the existing types of compression (generic or tailored compression) are requested. It can be selected at the system level, the data class level, or both.

Activation at the system level consists of the following components:

- In addition to the existing `TAILORED` and `GENERIC` values, the new `zEDC REQUIRED` (`ZEDC_R`) and `zEDC PREFERRED` (`ZEDC_P`) values are available on the **`COMPRESS`** parameter found in IGDSMSxx member of SYS1.PARMLIB.

  The `zEDC PREFERRED` option has been selected in Example 4-1.

  Example 4-1 IGDSMSxx member

```
SMS ACDS(SMS.ACDS) COMMDS(SMS.COMMDS)
RLSINIT(YES)
RLS_MAX_POOL_SIZE(500)
RLS_MAXCFFEATURELEVEL(Z)
TVSNAME(&TVSID1.)
TV_START_TYPE(WARM)
PDSESHARING(EXTENDED)
PDSE_RESTARTABLE_AS(YES)
HONOR_DSNTYPE_PDSE(YES)
SUPPRESS_SMSMSG(NO,IGD17054I,IGD17227I,IGD17395I)
PS_EXT_VERSION(2)
SAM_USE_HPF(YES)
MAXGENS_LIMIT(5)
PDSE_VERSION(2)
COMPRESS(ZEDC_P)
```

- The new **`COMPRESS`** parameter values behave in the following ways:
  - `ZEDC_R` specifies that the data set must be compressed using zEDC. With this option, 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.
  - `ZEDC_P` specifies that the data set be compressed using zEDC compression. However, the system does not fail the allocation request:
    - If the zEDC function is not supported by the system, it creates a tailored compressed data set.
    - If the minimum allocation amount requirement is not met, it creates a non-compressed extended format data set.

    For details about the `ZEDC_R` and `ZEDC_P` values, see *z/OS MVS Initialization and Tuning Reference,* SA23-1380.

    Note: The minimum allocation amount requirement is 5 megabytes (MB) primary allocation, or 8 MB primary if no secondary is specified.

    The IGDSMSxx member can be activated using the **`SET SMS=xx`** command, or by running an IPL on each LPAR where the feature is scheduled to become active.

Activation on the data class level also provides new components. In addition to the existing `Tailored` (`T`) and `Generic` (`G`) values, new `zEDC Required` (`ZR`) and `zEDC Preferred` (`ZP`) values will be available on the **`COMPACTION`** option in the DFSMS data class. When **`COMPACTION=Y`** in the data class, the system level is used.

For details about the ZR and ZP values, see *DFSMSdfp Storage Administration,* SC23-6860.

To activate on the data set level, we took the following steps:

1. Figure 4-3 shows Page 2 of 5 of the DFSMS Data Class Display panel. In our example, we defined a new Data Class, COMPZEDC, with the **COMPACTION** parameter set to `ZP`. Note that the **Data Set Name Type** parameter has to be set to `EXTENDED`.

```
Panel  Utilities  Scroll  Help

                              DATA CLASS DISPLAY                   Page 2 of 5
Command ===>

CDS Name  . . . . . : SMS.SCDS
Data Class Name . . : COMPZEDC

Data Set Name Type  . . . . . : EXTENDED
  If Extended . . . . . . . . : PREFERRED
  Extended Addressability . . : NO
  Record Access Bias  . . . . : USER
  RMODE31 . . . . . . . . . . :
Space Constraint Relief . . . : NO
  Reduce Space Up To (%)  . . :
  Dynamic Volume Count  . . . :

Compaction  . . . . . . . . . : ZP
Spanned / Nonspanned  . . . . :



Use UP/DOWN Command to View other Panels;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 4-3 Data Class Display panel: Page 2

2. Although generic and tailored compressed data sets can be defined as extended format version 1 or version 2 data sets, zEDC compressed data sets are defined as extended format version 2 data sets, regardless of the user's specification. User specification in data class, job control language (JCL), or SYS1.PARMLIB has no effect for this type of data set.

   Extended format version 2 (EF V2) data sets are new in IBM z/OS V2.1. The EF V2 format has been created to enable DFSMSdss support for IBM FlashCopy® when copying sequential, non-striped, multivolume EF V2 data sets.

   Restriction: There is a minor incompatibility between V1 and V2. Force end-of-volume (FEOV) is not supported on output for V2 data sets. The use of FEOV results in abnormal end of task (abend) 737-48.

3. In our test environment, we had to modify the DFSMS automatic class selection (ACS) Data Class routine so that a Data Class provided by the allocation routines is honored. See Example 4-2.

Example 4-2 Extract of ACS Data Class routine

```
/* -------------------------- START DC LOGIC ------------------------ */
SELECT
/* ---- KEEP ASSIGNED DATACLASS IF PROVIDED WITH THE ALLOCATION ----- */
WHEN (&DATACLAS NE '')
  DO
    SET &DATACLAS = &DATACLAS
    EXIT
  END
```

4. As a next step, we modified the ACS Storage Class routine so that we were able to provide specific data set patterns for the zEDC-compressed files. See Example 4-3.

Example 4-3 Extract of ACS Storage Class routine

```
/* -------------------------- START FILTERLIST ---------------------- */
FILTLIST MANAGED INCLUDE(PBRES*.ZEDC.**,
                         PBRES*.ZCOMP.**,
```

5. Because we wanted all zEDC compressed files to reside in a specific Storage Group, we also defined a new Storage Group, COMPZEDS. See the definitions in Figure 4-4.

```
 Panel  Utilities  Scroll  Help

                       POOL STORAGE GROUP DISPLAY                 Page 1 of 2
 Command ===>

 CDS Name  . . . . . : ACTIVE
 Storage Group Name  : COMPZEDS

 Description  : STORAGE GROUP FOR ZEDC COMPRESSED FILES

 Auto Migrate . . . . . . . . : NO
 Auto Backup  . . . . . . . . : NO
 Auto Dump  . . . . . . . . . : NO
 Overflow . . . . . . . . . . : NO
 Migrate Sys/Sys Group Name . :
 Backup Sys/Sys Group Name  . :
 Dump Sys/Sys Group Name  . . :
 Extend SG Name . . . . . . . :
 Copy Pool Backup SG Name . . :
 Dump Class . . . . . . . . . :
 Dump Class . . . . . . . . . :

 Use DOWN Command to View the next Page;
 Use HELP Command for Help; Use END Command to Exit.
```

Figure 4-4 Definition of new Storage Group COMPZEDS

6. After defining a new Storage Group, we modified the ACS Storage Group routine, so that the selection criteria for the newly defined data set pattern became valid for storage management subsystem (SMS)-managed volumes. See an excerpt in Example 4-4.

Example 4-4 partial extract of ACS Storage Group routine

```
SELECT
/* ----------- ASSIGN STORAGE GROUP TO ZEDC COMPRESSED FILES -------- */
 WHEN ( &DATACLAS = 'COMPZEDC' )
   DO
     SET &STORGRP = 'COMPZEDS'
     EXIT
   END
/* ------------------------------------------------------------------ */
```

4.3 Work with zEDC compressed files

With an environment ready to handle zEDC compressed files, we set up a series of tests to verify the function. First, we created some uncompressed sequential files. As Input we used the following libraries:

- SYS1.LINKLIB (DSORG=PO,RECFM=U,LRECL=0,BLKSIZE=32760)
- SYS1.LPALIB (DSORG=PO,RECFM=U,LRECL=0,BLKSIZE=32760)
- SYS1.PARMLIB (DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=23440)
- SYS1.PROCLIB (DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=23440)
- SYS1.MACLIB (DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=27920)

By using the TSO **XMIT** command, we created sequential files that we used as input for the tests using standard IBM utilities.

DFSMS identifies compressed data sets by using a dictionary token. The dictionary token identifies the type of compression. The values of the first two bytes of the token are shown in Table 4-1.

Table 4-1 Dictionary tokens

Token	Value # 1	Value # 2	Value # 3	Value # 5	Hex Value
Generic Token	.10.	.000	....	0000	X'4000'
Tailored Token	.11.	.xxxx	....	0000	X'6x00'
zEDC Token	.11.	.000	....	0001	X'6001'
Rejection Token	1...	....	....	....	X'8000'

4.3.1 BSAM files

To test zEDC with BSAM, we performed the following steps:

1. As a first test, we used IEBGENER to read and write the files we created. We set IEBGENER to use BSAM as access method. See example JCL in Example 4-5.

Example 4-5 JCL to run IEBGENER with BSAM

```
//JOB1      JOB ............
//F1        EXEC PGM=IEBGENR,REGION=0M
//SYSPRINT DD  SYSOUT=*
//SYSUT1    DD DISP=SHR,NCP=18,
//          DSN=PBRES2.NOCOMP.NOREC.BAM301.LINKLIB
//SYSUT2    DD DISP=(,CATLG),NCP=18,
//          DSN=PBRES2.ZCOMP.BAM301.F1B,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//          DATACLAS=COMPZEDC,
//          SPACE=(CYL,(20,10),RLSE),UNIT=(3390,4)
//SYSIN     DD DUMMY
```

2. After the test run, we compared the input and the output data sets. By looking at the output data set, we confirmed that DFSMS and the zEDC Express feature where properly configured. Figure 4-5 shows the input file.

```
Data Set Information
Command ===>
                                                                 More:     +
Data Set Name . . . . : PBRES2.NOCOMP.NOREC.BAM301.LINKLIB

General Data                          Current Allocation
 Management class . . : **None**       Allocated cylinders : 189
 Storage class  . . . : **None**       Allocated extents . : 18
  Volume serial . . . : WORKW3 +
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS            Current Utilization
  Record format . . . : FB             Used cylinders  . . : 189
  Record length . . . : 80             Used extents  . . . : 18
  Block size  . . . . : 3120
  1st extent cylinders: 20
  Secondary cylinders : 10            Dates
  Data set name type  :                Creation date . . . : 2014/11/06
                                       Referenced date . . : 2014/11/07
                                       Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 4-5 Shows the allocation details for the input file

3. Figure 4-6 shows the output file.

```
Data Set Information
Command ===>
                                                                More:     +
Data Set Name . . . . : PBRES2.ZCOMP.BAM301.F1B

General Data                       Current Allocation
 Management class . . : **None**    Allocated cylinders : 75
 Storage class  . . . : MANAGED     Allocated extents . : 7
  Volume serial . . . : #@$#Z2 +
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS         Current Utilization
  Record format . . . : FB          Used cylinders  . . : 75
  Record length . . . : 80          Used extents  . . . : 7
  Block size  . . . . : 32720
  1st extent cylinders: 20
  Secondary cylinders : 10         Dates
  Data set name type  : EXTENDED    Creation date . . . : 2014/11/07
                                    Referenced date . . : 2014/11/07
                                    Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 4-6 Shows the allocation details for the output file

As shown in Figure 4-6, the newly created output file was allocated as an extended-format sequential data set. DFSMS honored the correct Data Class, and the data set was allocated in the correct Storage Group. The output data set is compressed by a >2:1 ratio.

4. A **LISTCAT** of the compressed output file also showed the expected result (Example 4-6).

Example 4-6 Sample LISTCAT output

```
LISTCAT ENTRIES (PBRES2.ZCOMP.BAM301.F1B) ALL
NONVSAM ------- PBRES2.ZCOMP.BAM301.F1B
    IN-CAT --- UCAT.V#@$#M1
    HISTORY
     DATASET-OWNER-----(NULL)   CREATION--------2014.311
     RELEASE----------------2   EXPIRATION------0000.000
    ACCOUNT-INFO-----------------------------------(NULL)
    SMSDATA
     STORAGECLASS ----MANAGED   MANAGEMENTCLASS---(NULL)
     DATACLASS ------COMPZEDC   LBACKUP ---0000.000.0000
    VOLUMES
       VOLSER------------#@$#Z3     DEVTYPE------X'3010200F'
FSEQN------------------0
    ASSOCIATIONS--------(NULL)
    ATTRIBUTES
      VERSION-NUMBER---------2
      STRIPE-COUNT-----------1

ACT-DIC-TOKEN----X'60010000000400000000000000000000000000000000000000000000000
000000000000'
       COMP-FORMT      EXTENDED
```

4.3.2 QSAM files

To test zEDC with QSAM, we performed the following steps:

1. We then proceeded to read/write the same files we created before using IEBDG and QSAM as access method. See the sample JCL in Example 4-7.

Example 4-7 JCL to run IEBDG with QSAM

```
//*TESTJOB JOB ......
//F1       EXEC PGM=IEBDG,REGION=0M
//SYSPRINT DD    SYSOUT=*
//SYSUT1   DD DISP=SHR,BUFNO=18,
//         DSN=PBRES2.NOCOMP.NOREC.BAM301.LINKLIB
//SYSUT2   DD DISP=(,CATLG),BUFNO=18,
//         DSN=PBRES2.ZCOMP.BAM301.F1Q,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0),
//         DATACLAS=COMPZEDC,
//         SPACE=(CYL,(20,10),RLSE),UNIT=(3390,4)
//SYSIN    DD *
  DSD    OUTPUT=(SYSUT2),INPUT=(SYSUT1)
  CREATE INPUT=SYSUT1
  END
//*
```

2. Again, we compared the input data set (Figure 4-7) and the output data set (Figure 4-8 on page 63).

```
Data Set Information
Command ===>
                                                                More:     +
Data Set Name . . . . : PBRES2.NOCOMP.NOREC.BAM301.LINKLIB

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 189
 Storage class  . . . : **None**        Allocated extents . : 18
  Volume serial . . . : WORKW3 +
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used cylinders  . . : 189
  Record length . . . : 80              Used extents  . . . : 18
  Block size  . . . . : 3120
  1st extent cylinders: 20
  Secondary cylinders : 10             Dates
  Data set name type  :                 Creation date . . . : 2014/11/06
                                        Referenced date . . : 2014/11/07
                                        Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 4-7 Input QSAM data set

```
Data Set Information
Command ===>
                                                                More:     +
Data Set Name . . . . : PBRES2.ZCOMP.BAM301.F1Q

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 75
 Storage class  . . . : MANAGED         Allocated extents . : 7
  Volume serial . . . : #@$#Z2 +
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used cylinders  . . : 75
  Record length . . . : 80              Used extents  . . . : 7
  Block size  . . . . : 32720
  1st extent cylinders: 20
  Secondary cylinders : 10             Dates
  Data set name type  : EXTENDED        Creation date . . . : 2014/11/06
                                        Referenced date . . : 2014/11/06
                                        Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 4-8 Output QSAM data set

You notice the compression value is similar to BSAM.

4.3.3 Dumps

The system can produce several types of dumps. In our tests, we used a supervisor call (SVC) dump.

An SVC dump provides a representation of the virtual storage for the system when an error occurs. Typically, a system component requests the dump from a recovery routine when an unexpected error occurs. However, an authorized program or the operator can also request an SVC dump when diagnostic dump data is needed to solve a problem. For details, see the chapter about SVC dumps in *z/OS MVS Diagnosis: Tools and Service Aids*, GA32-0905.

SVC dump processing supports automatic allocation of dump data sets at the time the system writes the dump to direct access storage device (DASD). Automatically allocated dump data sets can be allocated as SMS-managed or non-SMS-managed, depending on the volume serial number (VOLSER) or SMS classes defined on the **DUMP ADD** command. When the system captures a dump, it allocates a data set of the correct size from the resources that you specify.

See the chapter about choosing SVC dump data sets in z*/OS MVS Diagnosis: Tools and Service Aids*, GA32-0905 for DFSMS support of extended-format sequential data sets. Using extended-format sequential data sets, the maximum size of the dump can exceed the size allowed for non-SMS managed data sets.

Example 4-8 shows the current setup of the Dump Server at our test system.

Example 4-8 Dump Server set up

```
ISF031I CONSOLE PBRES2 ACTIVATED
D D
IEE852I 17.13.03 SYS1.DUMP STATUS 052
SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=000
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00005000M
AUTOMATIC ALLOCATION IS: ACTIVE
   NO SMS CLASSES DEFINED
   AVAILABLE DASD VOLUMES: #@$#W1
   NAME=DUMP.D&MON.&DAY..H&HR..&SYSNAM..&JOBNAME..S&SEQ.
      EXAMPLE=DUMP.D1110.H22.#@$A.#MASTER#.S00000
```

The current setup shows that Dump Server will use automatic allocation for the dump data sets, and the naming pattern starts with "DUMP". A check of the automatic class selection (ACS) routines reveals that data sets starting with "DUMP.**" get a storage class of MANAGED, and will be allocated in Storage Group SGNORM.

Complete the following steps:

1. In our case, we assign a Data Class of COMPZEDC to our dump data sets, so that they will be compressed using the zEDC Express feature. We had to modify the Storage Group ACS routine as well, because the originally assigned Storage Group (COMPZEDS) is too small for the dump data sets. Example 4-9 shows the modification applied to the Storage Group ACS routine.

   Example 4-9 Storage Group ACS routine to define the COMPZEDC storage group

   ```
   SELECT
   /* ----------- ASSIGN STORAGE GROUP TO ZEDC COMPRESSED FILES -------- */
    WHEN ( &DATACLAS = 'COMPZEDC' )
      DO
        SET &STORGRP = 'SGNORM'
   /*   SET &STORGRP = 'COMPZEDS'      */
        EXIT
      END
   ```

2. The DUMPDS ADD command is **DD ADD,SMS=(D=COMPZEDC)**, as shown in Example 4-10, and it modifies the Dump Server parameters.

   Example 4-10 Dump Server parameter modification to assign COMPZEDC class

   ```
   DD ADD,SMS=(D=COMPZEDC)
   IEE855I DUMPDS COMMAND RESPONSE
   DUMPDS COMMAND SYS1.DUMP DATA SET STATUS
     SMS CLASSES ADDED: (DATA=COMPZEDC,MGMT=,STOR=)
   D D
   IEE852I 10.03.08 SYS1.DUMP STATUS 457
   SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=000
   CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00005000M
   AUTOMATIC ALLOCATION IS: ACTIVE
      AVAILABLE SMS CLASSES: (DATA=COMPZEDC,MGMT=,STOR=)
      AVAILABLE DASD VOLUMES: #@$#W1
      NAME=DUMP.D&MON.&DAY..H&HR..&SYSNAM..&JOBNAME..S&SEQ.
         EXAMPLE=DUMP.D1111.H15.#@$A.#MASTER#.S00000
   ```

3. The address space chosen for this example was IOSAS. We set up a SYS1.PARMLIB IEADMCxx member with the contents shown in Example 4-11.

Example 4-11 Address space for SAN Volume Controller dump

```
TITLE=('FRANCO PINTO SAMPLE DUMP FOR ZEDC TEST')
JOBNAME=(IXGLOGR),
SDATA=(COUPLE,ALLNUC,LPA,LSQA,PSA,RGN,SQA,TRT,CSA)
```

4. Execution of the DUMPDS command resulted in the allocation of an SMS-managed, zEDC-compressed extended-format sequential data set with the attributes shown in Figure 4-9.

```
Data Set Information
Command ===>
                                                                  More:     +
Data Set Name . . . . : DUMP.D1108.H21.#@$A.#MASTER#.S00022

General Data                           Current Allocation
 Management class . . : **None**        Allocated tracks  . : 2,912
 Storage class  . . . : MANAGED         Allocated extents . : 1
  Volume serial . . . : #@$#Z3
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS             Current Utilization
  Record format . . . : FBS             Used tracks . . . . : 2,912
  Record length . . . : 4160            Used extents  . . . : 1
  Block size  . . . . : 29120
  1st extent tracks . : 2912
  Secondary blocks  . : 17214          Dates
  Data set name type  : EXTENDED        Creation date . . . : 2014/11/08
                                        Referenced date . . : 2014/11/08
                                        Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 4-9 Display of SMS-managed, zEDC compressed, extended-format sequential data set

The dump file, when run without the options for zEDC compression, proved to be larger by a factor of 6:1 (17,959 tracks rather than 2,912 tracks when using zEDC compression). Also, the elapsed time for the dump to be finished dropped from 7 seconds to approximately 2.24 seconds when using zEDC compression services.

4.4 Identifying candidates

In support of helping to determine if there are files that are candidates for zEDC, IBM provides the IBM System z Batch Network Analyzer (zBNA) tool.

It is a no-charge, Microsoft Windows-based, “as is” productivity tool. It is designed to analyze batch windows using SMF data. The tool is available for clients, IBM Business Partners, and IBM employees. It replaces the BWATOOL. zBNA provides you with PC-based, graphical, and test reports, including Gantt charts and support for alternate processors.

zBNA can help identify zEDC compression candidates for BSAM and QSAM data sets across specified time spans, like batch windows. It helps estimate utilization of a zEDC feature, and size the number of features needed.

zBNA generates a list of data sets by job that already do hardware compression and might be candidates for zEDC. With zBNA, you can also generate lists of data sets by job that might be zEDC candidates, but are not in extended format.

For more details, see Chapter 5, “IBM System z Batch Network Analyzer Tool” on page 73.

4.5 DB2 for z/OS data and zEDC

DB2 is a database management system that was originally based on the relational data model, and is now extended to include hybrid object-relational and XML models. Many clients use DB2 for applications that require good performance and high availability (HA) for large amounts of data. This data is stored in data sets that are directly associated to DB2 table spaces, and distributed across DB2 databases. Data in table spaces is often accessed through indexes; indexes are stored in index spaces.

DB2 active data is allocated to VSAM data sets. Backup data, such as image copies, can be either collected in sequential data sets or in VSAM data sets, if the FlashCopy option is used.

From the z platform point of view, DB2 collects its Instrumentation Facility Component ID (IFCID) data into SMF records 100, 101, and 102. See Chapter 3, “z/OS zEnterprise Data Compression and System Management Facilities” on page 31.

DB2 might cause the occasional abend or SVC dumps, these can be allocated to zEDC-enabled devices as described in 4.3.3, “Dumps” on page 63.

DB2 data is involved in the general system backup volume dump procedures where data might be replicated across sites. For more information, see Chapter 6, “zEDC and DFSMSdss” on page 87.

4.5.1 Virtual Storage Access Method DB2 data sets

DB2 system table spaces and index spaces and DB2 user table spaces and index spaces are allocated on Virtual Storage Access Method (VSAM) linear data sets. Active logs are allocated on VSAM entry-sequenced data sets (ESDS), and bootstrap data sets are allocated on VSAM ESDS and key-sequenced data set (KSDS).

Image copies obtained using FlashCopy are a direct image of the active data. Target copies are VSAM linear data sets like the source.

VSAM data set compression does not support the zEDC Express feature, but standard DB2 compression (using compression call (CMPSC) instruction and a data dictionary) can be used on data. DB2 also provides a proprietary software compression for indexes.

VSAM data sets can be compressed if Data Set Services (DSS) is used to make physical copies of volumes.

4.5.2 Non-VSAM DB2 data sets

In addition to the data table spaces, DB2 requires a group of traditional data sets, not associated to table spaces, that are used by DB2 to distribute the software product and its maintenance, and to help provide the appropriate high level of data availability: The back-up data sets.

Typical non-VSAM data sets in DB2 environments are data sets, such as partitioned data sets extended (PDSE) for the DB2 modules. They do not support zEDC. (Note that PDSE does not support any type of data compression.)

DB2 uses sequential data sets for standard image copy utility output and utility work areas. The log archive data sets are also sequential data sets. They are written by DB2 using QSAM and read using BSAM.

All sequential data sets are candidates for zEDC compression. This can be accomplished by allocating the output using a zEDC-enabled DFSMS Data Class.

DB2 log archive data sets scenario

Example 4-12 shows an extract of a DB2 Master address space job log showing that DFSMS has been defined to assign a Data Class with zEDC support for the COPY2 data set of its archive logs.

Example 4-12 DB2 master syslog

```
12.47.51 STC03225  DSNJ072E  -DB1A ARCHIVE LOG DATASET  021
   021             'DB1AA.ARCHLOG2.A0000027' HAS BEEN ALLOCATED TO NON-TAPE DEVICE AND
   021             CATALOGUED.  ZPARM CATALOG OPTION OF 'NO' HAS BEEN OVERRIDDEN.
.......
12.47.52 STC03225  DSNJ003I  -DB1A DSNJOFF3 FULL ARCHIVE LOG VOLUME  023
   023             DSNAME=DB1AA.ARCHLOG2.A0000027, STARTRBA=00000000000E71399000,
   023             ENDRBA=00000000000E759E8FFF, STARTTIME=00CE135FA0599E614200,
   023             ENDTIME=00CE135FBE4D9907C000, UNIT=TAPE, COPY2VOL=SBOXG0, VOLSPAN=00,
   023             CATLG=YES
........
DB1AA.ARCHLOG2.A0000027
ALTHOUGH VOLUME COUNT REQUIREMENTS COULD NOT BE MET
IGD17070I DATA SET DB1AA.ARCHLOG2.A0000027
ALLOCATED SUCCESSFULLY WITH 1 STRIPE(S).
IGD17160I DATA SET DB1AA.ARCHLOG2.A0000027
IS ELIGIBLE FOR COMPRESSION
IGD101I SMS ALLOCATED TO DDNAME (SYS00042)
        DSN (DB1AA.ARCHLOG2.A0000027                     )
        STORCLAS (DB1AARCH) MGMTCLAS (MCDB22) DATACLAS (EXTZEDC)
        VOL SER NOS= SBOXG0
```

Figure 4-10 and Figure 4-11 show the data set characteristics for the two DB2 archive log data sets for the same time period.

```
Data Set Information
Command ===>
                                                                  More:     +
Data Set Name . . . . : DB1AA.ARCHLOG1.A0000027

General Data                           Current Allocation
 Management class . . : MCDB22          Allocated blocks  . : 3,000
 Storage class  . . . : DB1AARCH        Allocated extents . : 1
  Volume serial . . . : SBOXG1 +
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used blocks . . . . : 3,000
  Record length . . . : 4096            Used extents  . . . : 1
  Block size  . . . . : 24576
  1st extent blocks . : 3000
  Secondary blocks  . : 180            Dates
  Data set name type  :                 Creation date . . . : 2014/11/18
                                        Referenced date . . : 2014/11/18
                                        Expiration date . . : 2042/04/04
  SMS Compressible  . : NO
```

Figure 4-10 Standard archive log data set

```
Data Set Information
Command ===>
                                                                  More:     +
Data Set Name . . . . : DB1AA.ARCHLOG2.A0000027

General Data                           Current Allocation
 Management class . . : MCDB22          Allocated tracks  . : 376
 Storage class  . . . : DB1AARCH        Allocated extents . : 1
  Volume serial . . . : SBOXG1 +
  Device type . . . . : 3390
 Data class . . . . . : EXTZEDC
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used tracks . . . . : 366
  Record length . . . : 4096            Used extents  . . . : 1
  Block size  . . . . : 24576
  1st extent tracks . : 376
  Secondary blocks  . : 180            Dates
  Data set name type  : EXTENDED        Creation date . . . : 2014/11/18
                                        Referenced date . . : 2014/11/18
                                        Expiration date . . : 2042/04/04
  SMS Compressible  . : YES
```

Figure 4-11 zEDC-enabled archive log data set

One archive log (DB1AA.ARCHLOG1.A0000027) is allocated without using zEDC compression services, the other (DB1AA.ARCHLO21.A0000027) is allocated using zEDC compression services, as confirmed by the Syslog in Example 4-12 on page 67.

Note that in the first example, the current allocation is displayed in blocks. Because the block size is 24,576 bytes, two blocks fit on one 3390-9 track of 56,669 Bytes. Figure 4-12 shows the allocated space in tracks for both files.

```
 Menu  Options  View  Utilities  Compilers  Help

 DSLIST - Data Sets Matching DB1AA.ARCHLOG*.A0000027                Row 1 of 2
 Command ===>                                                    Scroll ===> CSR

 Command - Enter "/" to select action                          Tracks %Used   XT
 -------------------------------------------------------------------------------
          DB1AA.ARCHLOG1.A0000027                                1500  100     1
          DB1AA.ARCHLOG2.A0000027                                 376   97     1
 ***************************** End of Data Set list ****************************
```

Figure 4-12   Track allocation for DB2 archlog files

Notice in this sample the reduction in tracks used, from 1500 down to 366 for the zEDC compressed data set.

To get the DB2 archlog data sets zEDC compressed, we had to change the DFSMS ACS routines. Example 4-13 shows an excerpt of the data class selection routine.

Example 4-13   SMS data class

```
.................
/*--------- SET NEW FILTERLIST FOR DB1AA ARCHLOG FILES ---------------*/
FILTLIST DB1AA2       INCLUDE(DB1AA.ARCHLOG2.**)
/*--------------------------------------------------------------------*/
................
/*--------- SET DATACLASS EXTZEDC FOR DB1AA ARCHLOG FILES ------------*/
  WHEN (&DSN EQ &DB1AA2)
    DO
      SET &DATACLAS EQ 'EXTZEDC'
    END
/*--------------------------------------------------------------------*/
................
```

DB2 Image Copy data sets scenario

Our test table space contains the DB2 trace descriptions (4 columns). Rows were repetitively loaded to reach a total of 4,675,392. The table space was created with and without **COMPRESS YES** in the Data Definition Language (DDL).

Example 4-14 shows the standard Image Copy JCL, which copies the data to a sequential data set.

Example 4-14 Standard Image Copy

```
//DB2R2IFC  JOB (999,POK),'FELIPE',CLASS=A,
// MSGCLASS=T,NOTIFY=&SYSUID,REGION=0M
/*JOBPARM S=SC63,L=9999
//PROCLIB  JCLLIB ORDER=DB1AM.PROCLIB
//LOAD  EXEC DSNUPROC,SYSTEM=DB1A,
//             LIB='DB1AT.SDSNLOAD',
//             UID='LOADPP' UTPROC='PREVIEW'
//*SNUPROC.SYSREC DD DISP=SHR,DSN=DB1AT.SDSNIVPD(DSNWMSGS)
//DSNUPROC.SYSREC DD DISP=SHR,DSN=FELIPE.DB1A.UNLD.DSN8D11A.TRACETS
//DSNUPROC.SYSIN DD *
          TEMPLATE COPY DSN 'DB2R2.&DB..&TS..T&TIME.'
           DISP (NEW,CATLG,DELETE)
           UNIT SYSDA
           SPACE (50,50) CYL
          TEMPLATE UT1  DSN 'DB2R2.&DB..&TS..SYSUT1'
           DISP (NEW,DELETE,DELETE)
           UNIT SYSDA
```

Example 4-15 shows the JCL used to copy the data to a zEDC compressed extended-format sequential data set.

Example 4-15 Image Copy directed to the zEDC enabled storage group

```
//DB2R2GLW  JOB (999,POK),'FELIPE',CLASS=A,
//*       RESTART=STEPNAME, <== FOR RESTART REMOVE * AND ENTER STEP NAME
// MSGCLASS=T,NOTIFY=&SYSUID,REGION=0M
/*JOBPARM S=SC63,L=9999
//COPY1 EXEC DSNUPROC,SYSTEM=DB1A,
//             LIB='DB1AT.SDSNLOAD',
//             UID=''
//DSNUPROC.SYSCOPY DD DSN=FELIPE.DB1A.IC.DSN8D11A.TRACETS.EXTZEDC.COMP,
//             DISP=(NEW,CATLG),
//             SPACE=(CYL,(900,900),RLSE),
//*            UNIT=SYSDA,VOL=SER=(BOX008,BOX009)
//             UNIT=SYSDA,DATACLAS=EXTZEDC
//DSNUPROC.SYSIN  DD  *
```

Table 4-2 summarizes the DASD allocation and execution time of image copies in four cases:

- DB2 compressed and zEDC not compressed
- DB2 not compressed and zEDC not compressed
- DB2 compressed and zEDC compressed
- DB2 not compressed and zEDC compressed

We notice that, for this test case, DB2 had compressed the table space by about 50%.

zEDC is capable of further compressing the DB2 compressed data by another 50%, with the best case being the full zEDC compression.

Table 4-2 zEDC compression of Image Copies

DB2 Compressed	zEDC Compressed	Tracks	Extents	Execution time (min.)
Y	N	46,050	7	0.30
N	N	97,485	8	0.41
Y	Y	21,750	2	0.22
N	Y	17,670	2	0.50

The compressed image copies can be migrated by DFSMS hierarchical storage manager (DFSMShsm) to tape. DFSMS data set services (DFSMSdss) has to be defined as the data mover in DFSMShsm.

Table 4-3 evidences the space allocation using the various compression methods when DFSMShsm migrates the image copy files. Similar results could be achieved when migrating the DB2 archive log files.

Table 4-3 Space allocated using DFSMShsm migration

zEDC used during image copy	DB2 compression used during image copy	Number of tracks allocated on DASD	Number of tracks allocated when migrated to ML1[a]	Number of blocks allocated when migrated to ML2[a]	Number of tracks allocated when migrated to ML2[a]
N	Y	46,050	22,681	63,535	18,371
N	N	97,485	19,315	63,536	18,371
Y	Y	21,750	22,317	63,535	18,371
Y	N	17,670	18,122	61,160	17,684

a. Note that zEDC compression has to be enabled in DFSMShsm. For more information, see 7.2.1, “Specifying when compression with zEDC should be done” on page 111.

5

IBM System z Batch Network Analyzer Tool

This chapter introduces the IBM System z Batch Network Analyzer (zBNA) tool, and gives information about how to obtain this no initial charge tool. It also provides an example about how to use the tool to identify jobs, and basic sequential access method (BSAM) and queued sequential access method (QSAM) data sets, that are IBM zEnterprise Data Compression (zEDC) Express compression candidates.

This chapter describes the following topics:

- Introduction to zBNA
- Installation of the zBNA tool
- Example of zBNA zEDC Express analysis

© Copyright IBM Corp. 2015. All rights reserved.

5.1 Introduction to zBNA

IBM zBNA is a no-charge, as-is tool that can be used to analyze a single batch window of user-defined length. It reads System Management Facilities (SMF) records, analyzes how processor capacity is being used, and projects what work would be most sensitive to changes in engine speed. It is personal computer (PC)-based, and provides graphical and test reports, including Gantt charts.

zBNA version 1.3 provides a means of estimating the number of jobs and BSAM/QSAM data sets that might be eligible for compression using the zEDC Express feature, and helps determine the number of features needed.

5.1.1 How to obtain zBNA

The zBNA tool is available to clients, IBM Business Partners, and IBM employees:

- IBM clients can obtain zBNA and other Capacity Planning Support (CPS) tools from the following website:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132

- IBM Business Partners can obtain zBNA and other Capacity Planning Support (CPS) tools from the following website:

  https://www.ibm.com/partnerworld/wps/servlet/mem/ContentHandler/tech_PRS5133

- IBM employees can obtain zBNA and other CPS tools from the IBM intranet:

  http://w3.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5126

5.1.2 Identification of zEDC Express compression candidates

Based on client-provided SMF records, the zBNA tool can be used to identify jobs and BSAM and QSAM data sets that are zEDC Express compression candidates, across a specified time window (typically a batch window).

zBNA is able to generate a list of data sets by jobs:

- Jobs that already perform hardware compression and might be candidates for zEDC Express
- Jobs that might be zEDC Express candidates, but are not in extended format

Furthermore, zBNA estimates the use of a zEDC Express feature, and the number of features needed.

5.2 Installation of the zBNA tool

In this section, we show a step-by-step example of zBNA tool installation for using the zBNA to identify BSAM/QSAM data sets that are zEDC Express compression candidates across a specified time window.

Section 5.3, “Example of zBNA zEDC Express analysis” on page 80 provides an example of how to analyze the output from the zBNA tool.

The zEDC tool is a PC-based productivity tool.

For details about minimum requirements and installation of the zBNA tool, obtain the current version of the *IBM System z Batch Network Analyzer User's Guide* available in a Portable Document Format (PDF) file from the zBNA site:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132

5.2.1 Input data gathering

The zBNA tool accepts SMF Types 14, 15, 30, 42, 70, and 113 data extracted using the CP3KEXTR tool to analyze a single batch window of user-defined length. The program is focused on batch jobs, and ignores records that are not batch jobs.

The CP3KEXTR program is used to read SMF records. It produces Enterprise Data Files (EDF) that are read into Processor Capacity Reference for IBM System z (zPCR). zPCR provides capacity relationships for System z processors considering the following information:

- Logical partition (LPAR) configuration
- Secure Copy (SCP)/workload environment
- Use of specialty processors:
  - System z Application Assist Processor (zAAP)
  - System z Integrated Information Processor (zIIP)
  - Integrated Facility for Linux (IFL)
  - Integrated catalog facility (ICF)

CP3KEXTR is offered as a no initial charge application. The CP3KEXTR tool has to be run inside the IBM z/OS system, and the output has to be transferred in text format using File Transfer Protocol (FTP) to the PC where the zBNA tool is installed. The tool is available from the following website:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4229

> **Note:** SMF record Types 14 and 15 are required for the zEDC Express analysis.

For details about the input data gathering, obtain the current version of the *IBM System z Batch Network Analyzer User's Guide* available as a PDF file from the zBNA site:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132

5.2.2 Analyzing the output

The first step is to import the SMF70 (`.edf`) and z/OS SMF (`.dat`) files into the zBNA tool:

1. Click **File** → **Load Files**. Click **Browse for SMF70 file** and select a `.edf` file.
2. Click **Browse for z/OS SMF file** and select a `.dat` file. Click **Import** (Figure 5-1).

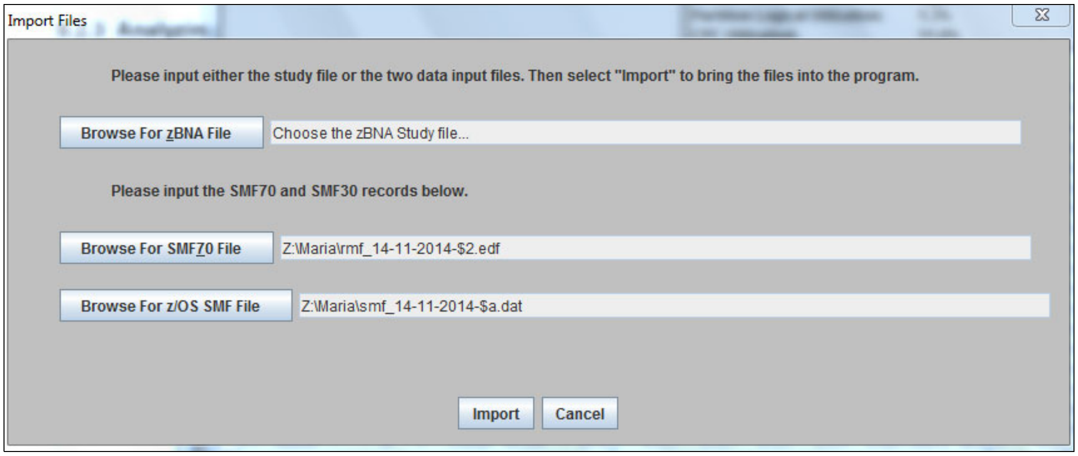


Figure 5-1 Import files

zEDC Top Data Sets

To start the zEDC analysis, use zBNA to add SMF Record Types 14 and 15 (Figure 5-2):

1. Click **Action** → **zEDC: Compression**.

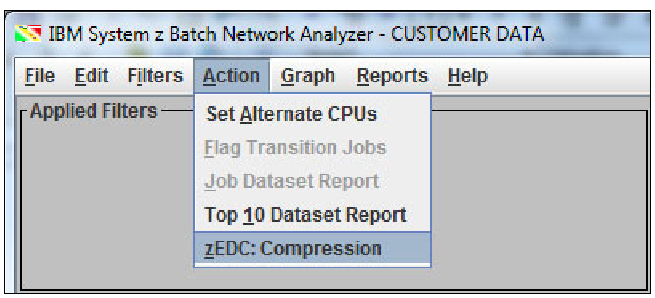


Figure 5-2 How to start the zEDC analysis

2. A list of the top zEDC Express candidate data sets are generated. See Figure 5-3.

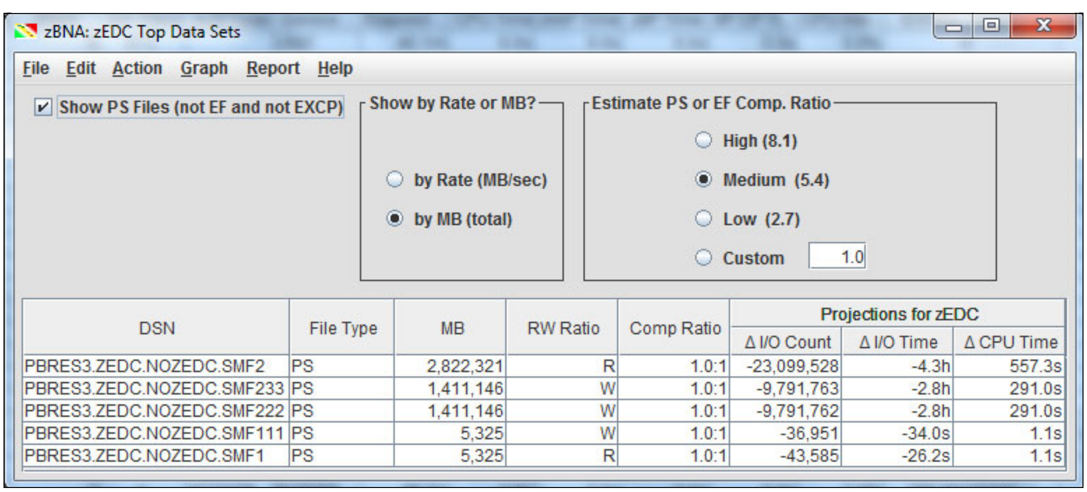


Figure 5-3 List of the top zEDC Express candidate data sets

The data sets are filtered by file type:

COMP Already compressed using System z hardware compression.

EF Extended format data set but not compressed.

PS Physical sequential not extended format.

zEDC Using the zEDC Express feature for compression.

zEDC data set analysis

After the zEDC top data sets list has been created, you have the option to create a graph of the zEDC top data sets, projected zEDC cards, central processing unit (CPU) savings, and input/output (I/O) count.

Graphical view of the zEDC Top Data Sets

You can get a graphical view of the zEDC top data sets:

1. Right-click a data set, and click **zEDC Dataset Analysis** to create the graph in Figure 5-4.

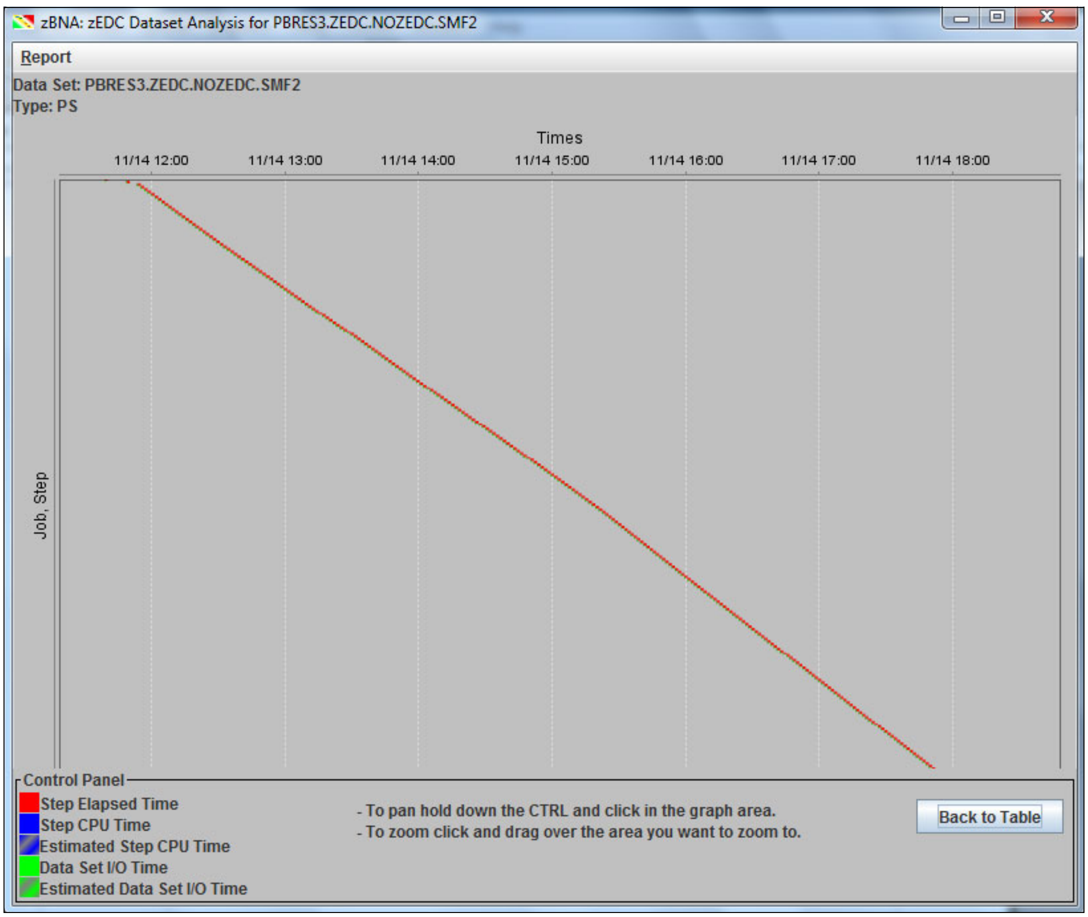


Figure 5-4 Graphical view of a selected zEDC top data set

The upper left of the panel shows the data set represented and its compression category.

The y-axis shows the corresponding job and step in the data set.

The control panel contains a description of the color scheme of the graph.

2. To zoom in on the graph, click and drag the cursor over the area you want to zoom in to. You can do this multiple times to zoom in and get the view you want. See Figure 5-5.

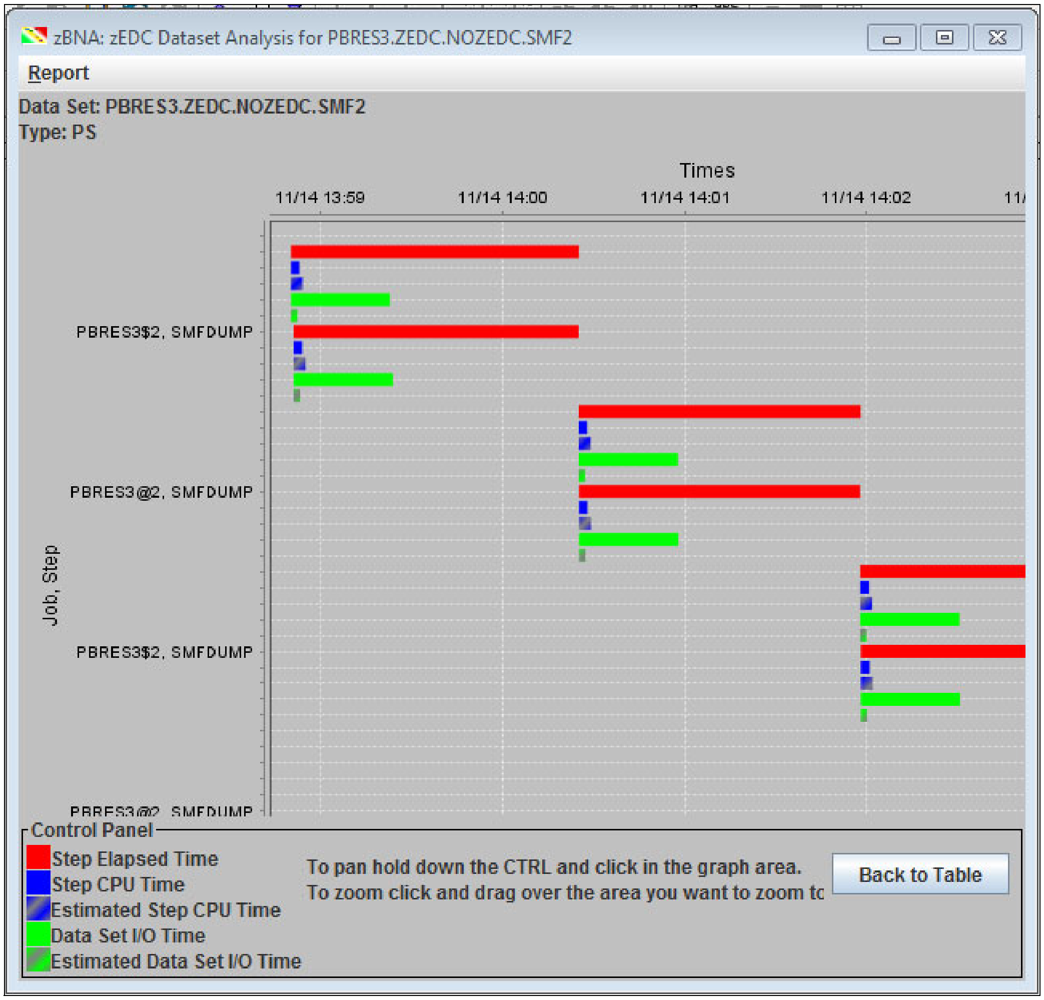


Figure 5-5 Zoom - Graphical view of a selected zEDC top data set

By running the cursor over each bar in the graph you get the number for step elapsed time, step CPU time, estimated CPU time, data set I/O time, and estimated data set I/O time in a pop-up box.

Projected zEDC cards

You can get a graphical view of the estimated number of zEDC features the system needs to support the workload for all BSAM/QSAM data sets:

1. Click **Graph** → **Projected zEDC cards**. The graph in Figure 5-6 on page 79 is created.
2. The x-axis shows the start and end hour of this data.

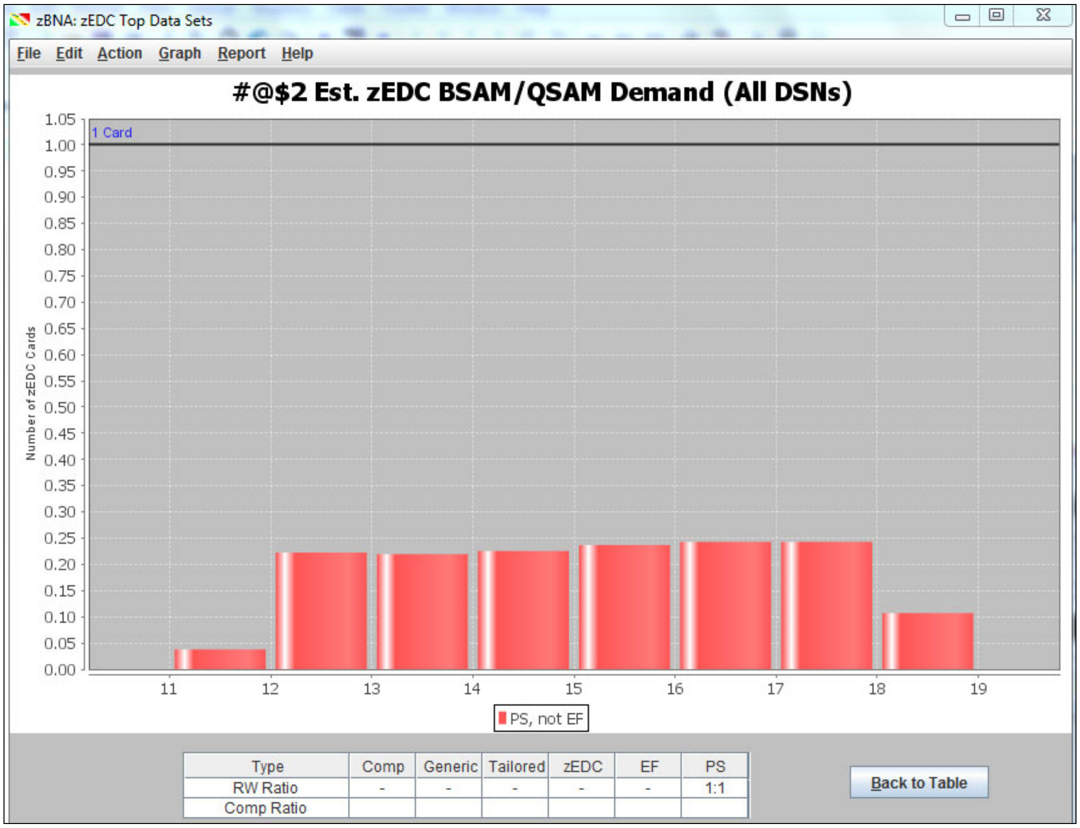


Figure 5-6 Projected zEDC cards

CPU savings

You can get a graphical view of the CPU savings and cost estimate when using the zEDC feature for BSAM/QSAM:

1. Click **Graph** → **Projected zEDC CPU Savings**. The graph in Figure 5-7 is created.

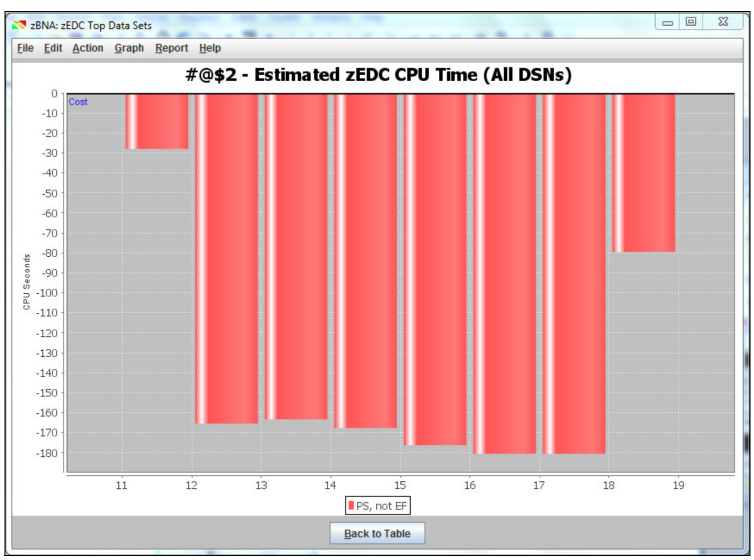


Figure 5-7 Estimated zEDC CPU savings

2. The CPU savings included in the zBNA analysis is *only* including DFSMS data sets that are using Hardware Data Compression (Generic or Tailored).

I/O Count

You can get a graphical view of the I/O savings estimate when using the zEDC feature for BSAM/QSAM:

1. Click **Graph** → **Projected zEDC I/O Count**. The graph in Figure 5-8 is created.

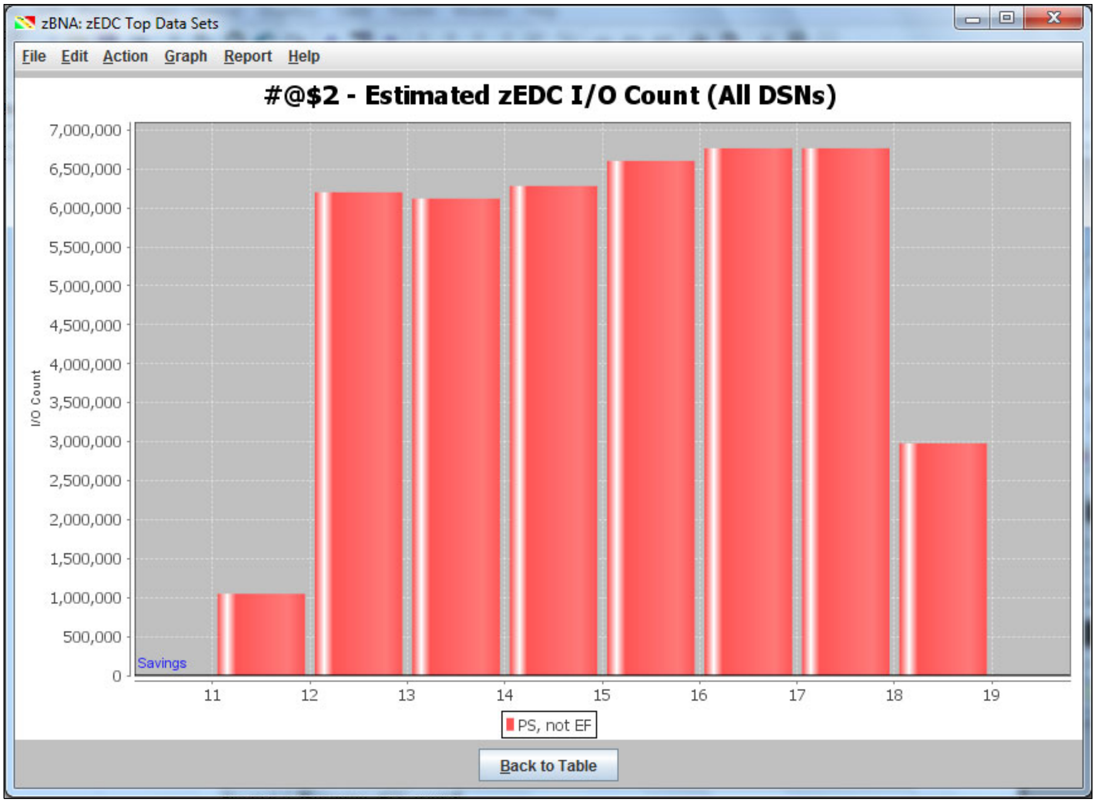


Figure 5-8 Estimated I/O savings

5.3 Example of zBNA zEDC Express analysis

In this section, we show a step-by-step analysis of the output from the zBNA tool, based on a test case. The example covers z/OS systems *without* and *with* the zEDC Express feature available.

Test case description

The test case was conducted to demonstrate how to analyze the output of the zBNA tool across a specified time window:

- Identifying BSAM/QSAM data sets that are zEDC Express candidates
- The potential I/O and CPU savings
- The number of zEDC features needed to support the workload

The test was conducted in a z/OS system (#@$2) *without* the zEDC feature available for the LPAR, and in a z/OS system (#@$A) *with* the feature available for the LPAR. Both systems are on the same zEC12. The purpose is to illustrate a *before and after zEDC* scenario.

In both cases, two parallel jobs were concurrently running, and we created a flow of 540 jobs using the same job name. The content of the input data sets consists of SMF records, and the jobs ran the SMF dump utility IFASMFDP.

Figure 5-9 shows the input data set used in the z/OS system *without* zEDC available for the LPAR.

```
 Data Set Information
Command ===>

Data Set Name . . . . : PBRES3.ZEDC.NOZEDC.SMF2

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 6,648
 Storage class  . . . : MANAGED         Allocated extents . : 7
  Volume serial . . . : TRA055 +
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : VBS             Used cylinders  . . : 6,648
  Record length . . . : 32760           Used extents  . . . : 7
  Block size  . . . . : 27998
  1st extent cylinders: 2000
  Secondary cylinders : 900            Dates
  Data set name type  :                 Creation date . . . : 2014/11/14
                                        Referenced date . . : 2014/11/14
                                        Expiration date . . : ***None***

  SMS Compressible  . : NO
```

Figure 5-9 Without zEDC data set characteristics

Figure 5-10 shows the job used in the z/OS system *without* zEDC available for the LPAR.

```
//PBRES3$2  JOB ACCNT#,PBRES3,NOTIFY=PBRES3,MSGLEVEL=(1,1)
/*JOBPARM SYSAFF=#@$2
//DEL1     EXEC PGM=IEFBR14
//DD       DD DSN=PBRES3.ZEDC.NOZEDC.SMF222,DISP=(MOD,DELETE),
//         SPACE=(CYL,1),UNIT=SYSDA
//SMFDUMP  EXEC PGM=IFASMFDP,REGION=0M
//DUMPIN   DD DISP=SHR,DSN=PBRES3.ZEDC.NOZEDC.SMF2
//DUMPOUT  DD DSN=PBRES3.ZEDC.NOZEDC.SMF222,UNIT=(3390,4),
//         DISP=(NEW,CATLG),SPACE=(CYL,(3000,500),RLSE)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  INDD(DUMPIN,OPTIONS(DUMP))
  OUTDD(DUMPOUT,TYPE(1:255))
/*
```

Figure 5-10 Without zEDC job sample

Figure 5-11 describes the input data set used in the z/OS system *with* zEDC available for the LPAR.

```
Data Set Information
Command ===>

Data Set Name . . . . : PBRES3.ZEDC.ZEDC.SMFAAA

General Data                        Current Allocation
 Management class . . : **None**     Allocated cylinders : 658
 Storage class  . . . : MANAGED      Allocated extents . : 1
  Volume serial . . . : TRA354 +
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS          Current Utilization
  Record format . . . : VBS          Used cylinders  . . : 658
  Record length . . . : 32767        Used extents  . . . : 1
  Block size  . . . . : 32760
  1st extent cylinders: 658
  Secondary cylinders : 300         Dates
  Data set name type  : EXTENDED     Creation date . . . : 2014/11/14
                                     Referenced date . . : 2014/11/15
                                     Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 5-11 With zEDC data set characteristics

Figure 5-12 lists the job used in the z/OS system *with* zEDC available for the LPAR.

```
//PBRES3$A  JOB ACCNT#,PBRES3,NOTIFY=PBRES3,MSGLEVEL=(1,1)
/*JOBPARM SYSAFF=#@$A
//DEL1     EXEC PGM=IEFBR14
//DD       DD DSN=PBRES3.ZEDC.ZEDC.SMFCCC,DISP=(MOD,DELETE),
//         SPACE=(CYL,1),UNIT=SYSDA
//SMFDUMP  EXEC PGM=IFASMFDP,REGION=0M
//DUMPIN   DD DISP=SHR,DSN=PBRES3.ZEDC.ZEDC.SMFAAA
//DUMPOUT  DD DSN=PBRES3.ZEDC.ZEDC.SMFCCC,UNIT=(3390,4),
//         DISP=(NEW,CATLG),SPACE=(CYL,(2000,300),RLSE),
//         DATACLAS=COMPZEDC
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  INDD(DUMPIN,OPTIONS(DUMP))
  OUTDD(DUMPOUT,TYPE(1:255))
/*
```

Figure 5-12 With zEDC job sample

Data set analysis

After you have loaded the output of the CP3KEXTR tool into the zBNA tool, it provides a list of the top zEDC Express candidate data sets. See Figure 5-13.

DSN	File Type	MB	RW Ratio	Comp Ratio	Projections for zEDC		
					Δ I/O Count	Δ I/O Time	Δ CPU Time
PBRES3.ZEDC.NOZEDC.SMF2	PS	2,822,321	R	1.0:1	-23,099,528	-4.3h	557.3s
PBRES3.ZEDC.NOZEDC.SMF233	PS	1,411,146	W	1.0:1	-9,791,763	-2.8h	291.0s
PBRES3.ZEDC.NOZEDC.SMF222	PS	1,411,146	W	1.0:1	-9,791,762	-2.8h	291.0s

DSN	File Type	MB	RW Ratio	Comp Ratio
PBRES3.ZEDC.ZEDC.SMFAAA	ZEDC	2,870,475	R	10.0:1
PBRES3.ZEDC.ZEDC.SMFCCC	ZEDC	1,437,532	W	10.0:1
PBRES3.ZEDC.ZEDC.SMFCDD	ZEDC	1,432,937	W	10.0:1

Figure 5-13 Top zEDC Express candidate data sets. Top: Without zEDC (#@$2) Bottom: With zEDC (#@$A)

For the z/OS system *without* zEDC available, the file type for the data sets is physical sequential not extended format (PS). This means that these data sets are candidates for compression with the zEDC. The first data set is read (R) and the next two are write (W). That is our test case: Two concurrent jobs at the same time.

In the three columns to the right, the projections are for the following data:

- The I/O count
- I/O time
- CPU time

The zBNA tool estimates that the data set SMF2 will save 23,099,528 in I/O count, and the I/O time will be reduced by 4.3 hours, if the zEDC is used for compression.

For the z/OS system *with* zEDC available, the file type is zEDC, which means the zEDC Express feature is used to compress the data sets. The compression ratio (Comp Ratio) for the data set SMFAAA is 10.0:1 when compressing with the zEDC. This is the uncompressed data size divided by the compressed data size. Figure 5-14 is a graphical view of the top zEDC Express candidate data sets, zoomed in on a selected job.

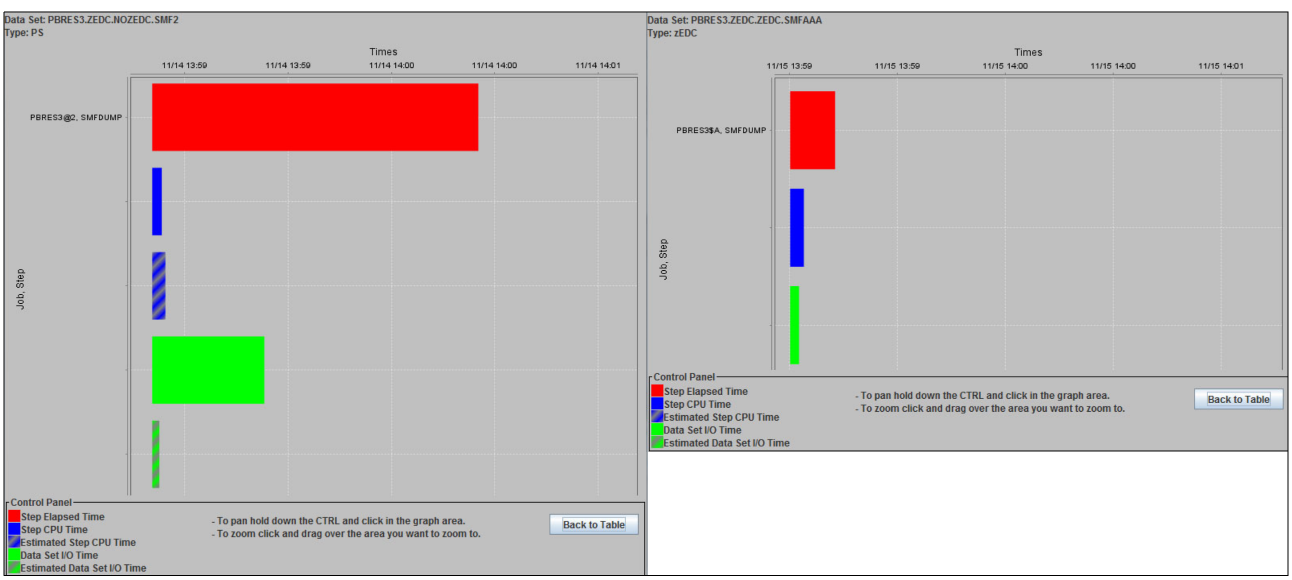


Figure 5-14 Graphical view - Top candidate data sets. Left: Without zEDC (#@$2) Right: With zEDC (#@$A)

If you compare the two graphs of the top candidate data sets, without and with zEDC in Figure 5-14 on page 83, they show that the elapsed time and I/O time is shorter when zEDC is used. The estimated CPU time is longer.

Now look at the graphs for the I/O counts, CPU savings, and projected zEDC cards.

I/O Count analysis

The graphs illustrate an overall strong reduction of elapsed time and I/O count for the same number of jobs when using the zEDC feature. See Figure 5-15.

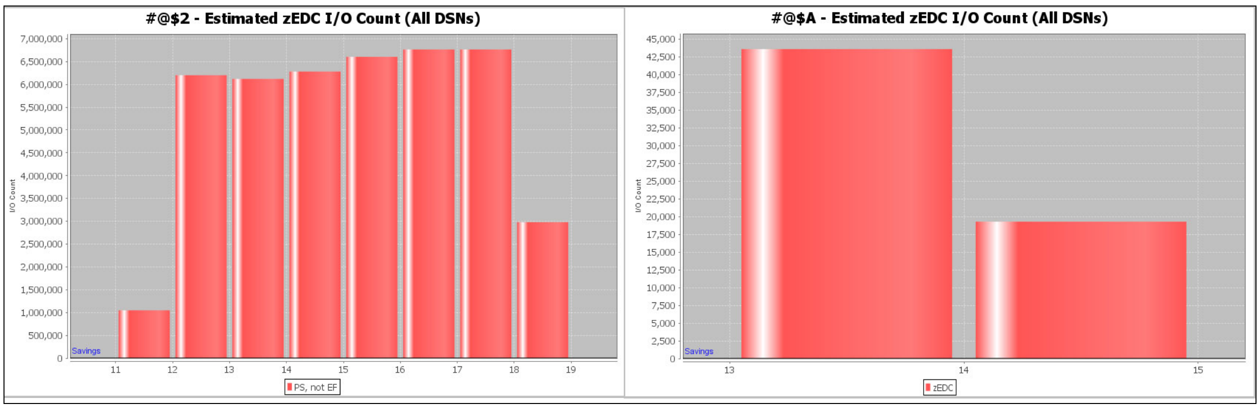


Figure 5-15 I/O savings. Left: Without zEDC (#@$2) - Right: With zEDC (#@$A)

CPU saving analysis

Overall, you can see that without the zEDC there is a CPU cost, although if you use the zEDC, you save some CPU use. See Figure 5-16. For our test case, the CPU savings is so small that the graph is slightly misleading.

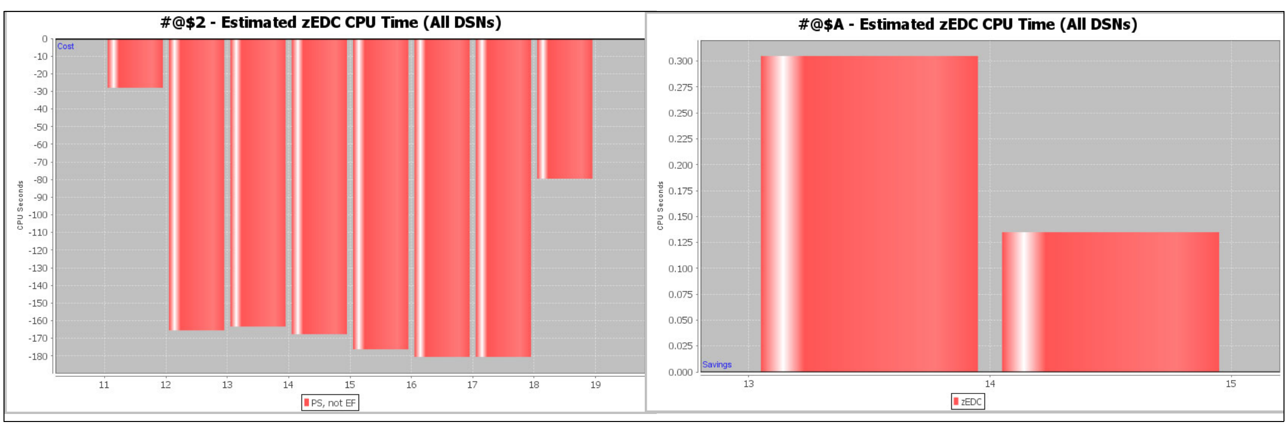


Figure 5-16 CPU savings. Left: Without zEDC (#@$2) Right: With zEDC (#@$A)

Projected zEDC cards analysis

zBNA estimates that from a capacity perspective, for the z/OS system without the card available, one card is enough. However, because the throughput is better in the z/OS system with zEDC, you can observe that one card is exceeded. See Figure 5-17. The enhanced throughput is something that you need to include in your considerations when using the zBNA tool to estimate the number of zEDC features that the system needs to support the workload.

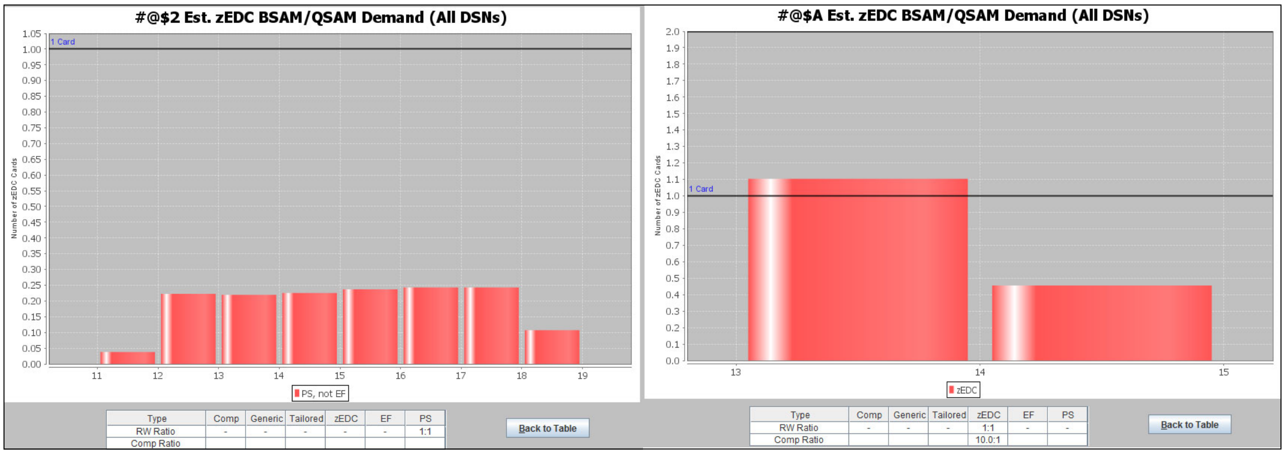


Figure 5-17 zEDC cards. Left: Without zEDC (#@$2) Right: With zEDC (#@$A)

Detailed overview

For a more detailed overview, Table 5-1 contains step elapsed time, step CPU time, data set I/O time, and data set I/O start sub-channel (SSCH) time for all three data sets, one read and two write, for both z/OS systems. The source is Figure 5-14 on page 83, the graphical view of the top zEDC Express candidate data sets, zoomed in on a selected job.

Table 5-1 Test case numbers

LPAR	Data set (DS)	Tracks	R or W	# of job	elapsed sec.	CPU sec.	Est. CPU sec.	DS I/O sec.	DS I/O SSCH	Est. DS I/O sec.	Est. DS I/O SSCH
no zEDC	NOZEDC.SMF2	99,720	R	530	**97.1**	2.8	3.8	**32.6**	**46,592**	2.1	2,944
no zEDC	NOZEDC.SMF222	99,720	W	265	**97.1**	2.7	3.8	**43.9**	**39,895**	3.2	2,944
no zEDC	NOZEDC.SMF233	99,720	W	265	**96.8**	2.7	3.8	**42.4**	**39,895**	3.1	2,944
zEDC	ZEDC.SMFAAA	9,870	R	540	**12.3**	3.8		**2.3**	**1,645**		
zEDC	ZEDC.SMFCCC	9,870	W	270	**12.3**	3.8		**3.8**	**1,646**		
zEDC	ZEDC.SMFCDD	9,870	W	270	**12.1**	3.8		**3.8**	**1,646**		

The elapsed time for the data set from the z/OS system without the zEDC available is 97.1 sec. compared with 12.3 sec. for the data set that has been compressed using zEDC.

Therefore, in our test case, the total elapsed time for all 540 jobs is reduced 7.9 times.

The result comes from the benefit of extended format zEDC compression that reads 3x as many physical tracks per SSCH and the compression ratio of 10.0:1 (see Figure 5-13 on page 83). You can see this benefit in the DS I/O SSCH number: 46,592/1,645 = 28.3.

Summary

The graph in Figure 5-14 on page 83 provides an overall view, and by looking into the numbers behind it, you can analyze the real benefits of using zEDC.

In our test case, the benefit of using the zEDC card is in the substantial I/O savings obtained by combining the compression ratio and the increase in physical track per SSCH.

6

zEDC and DFSMSdss

In this chapter, we describe the usage of Data Facility Storage Management Subsystem data set services (DFSMSdss) as the basic data mover for files. DFSMSdss is a direct access storage device (DASD) data and space management tool. DFSMSdss works on DASD volumes only in the IBM z/OS environment.

We describe how DFSMSdss uses the IBM zEnterprise Data Compression (zEDC) Express feature to reduce physical DASD space while working with sequential data sets.

We define the DFSMSdss parameters for the functions that explore the zEDC Express feature, and then show the usage of the DFSMSdss COPY, DUMP, and RESTORE functions with the zEDC Express feature in detail.

We also show the use of DFSMSdss to create zEDC compressed data on tape.

This chapter contains the following sections:

© Copyright IBM Corp. 2015. All rights reserved.

6.1 DFSMSdss functions that support zEDC Express

The following DFSMSdss tasks support data sets in zEDC Express compressed format:

- **CONSOLIDATE**
- **COPY**
- **DEFRAG**
- **DUMP**
- **RESTORE**
- **PRINT**

Note: As of today, when copying or restoring compressed format data sets, the type of compression used is carried along from the source. This is true whether the preallocated target is usable, or had to be scratched and reallocated. Also, DFSMSdss does not support copying a compressed format data set to a non-compressed format data set or vice versa.

DFSMSdss enables a user on z/OS V1R12 and V1R13 to **RESTORE** a compressed format sequential data set when the form of compression used was zEDC compression. Information indicating that the data set is in a compressed format is preserved during the **RESTORE**.

DFSMSdss fails logical data set **COPY** and **DUMP** operations of extended format data sets in the zEDC compressed format. A new reason code is added to the existing ADR778E error message indicating that a compressed format data set compressed with the zEDC form of compression is not supported on this release.

DFSMSdss fails logical data set **COPY** and **RESTORE** operations when a pre-allocated output data set is in the zEDC compressed format. A new reason code is added to the existing ADR285E error message indicating that a pre-allocated compressed format data set compressed with the zEDC form of compression is not supported on this release.

Program temporary fixes (PTFs) for zEDC use or software decompression have a fix category of IBM.Function.zEDC.

The following list describes the applicable authorized program analysis report (APARs):

- zEDC format sequential data set support:
  - OA42198
  - OA43817
- Partial Release Reporting error:
  - OA45229
- zEDC exploitation:
  - OA42238 contains PTFs for HDZ2210, HDZ1D10, HDZ1C10

6.2 DFSMSdss tasks and parameters with zEDC Express

This section includes the tests implemented for the various DFSMSdss tasks:

- **CONSOLIDATE**
- **COPY**
- **DEFRAG**
- **DUMP** and **RESTORE**
- **PRINT**

6.2.1 CONSOLIDATE

You can use the **CONSOLIDATE** command to consolidate the multi-extent data sets that are on a single volume, and that are not excluded from data movement. For eligible data sets that consist of contiguous extents in sequential order, DFSMSdss relocates eligible data set extents if contiguous free space exists on the volume to hold the extents.

Example 6-1 shows a sample **CONSOLIDATE** operation.

Example 6-1 CONSOLIDATE operation

```
//JOB1         JOB ........
//CONSOLID EXEC     PGM=ADRDSSU
//SYSPRINT     DD       SYSOUT=*
//DASD         DD       UNIT=3390,VOL=(PRIVATE,SER=#@$#Z3),DISP=OLD
//SYSIN        DD       *
 CONSOLIDATE -
  DATASET(INCLUDE -
  (**)) -
  PHYSINDDNAME(DASD)
/*
```

We established a separate Storage Group for the zEDC compressed files, then we ran the **CONSOLIDATE** operation for one volume in this Storage Group at a time. The results are shown in Example 6-2.

Example 6-2 Storage Group for the zEDC compressed files for CONSOLIDATE

```
PAGE 0001     5695-DF175  DFSMSDSS V2R01.0 DATA SET SERVICES     2014.316 10:27
 CONSOLIDATE -
  DATASET(INCLUDE -
  (**)) -
  PHYSINDDNAME(DASD)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'CONSOLIDATE'
ADR109I (R/I)-RI01 (01), 2014.316 10:27:26 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2014.316 10:27:26 EXECUTION BEGINS
ADR806I (001)-DFRGD(01), RELOCATED EXTENTS WILL BE COPIED USING A FAST REPLICATION FUNCTION
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR PBRES3.ZEDC.SMF1, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR PBRES2.ZEDC.VSAM.DUMP, 01
ADR260I (001)-DFRGD(01), EXTENTS REDUCED FROM 002 TO 001 FOR PBRES2.ZEDC.DB1A.IC.DSN8D11A.TRACETS.NOCOMP2
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR DUMP.D1108.H20.#@$A.#MASTER#.S00013, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR PBRES1.ZEDC.SMF2, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR PBRES3.ZEDC.SMF22, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR DUMP.D1108.H20.#@$A.#MASTER#.S00014, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR DUMP.D1108.H20.#@$A.#MASTER#.S00015, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR DUMP.D1108.H20.#@$A.#MASTER#.S00017, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR DUMP.D1108.H21.#@$A.#MASTER#.S00022, 01
ADR454I (001)-DFRGD(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          PBRES3.ZEDC.SMF1
                          PBRES2.ZEDC.VSAM.DUMP
                          PBRES2.ZEDC.DB1A.IC.DSN8D11A.TRACETS.NOCOMP2
                          DUMP.D1108.H20.#@$A.#MASTER#.S00013
                          PBRES1.ZEDC.SMF2
                          PBRES3.ZEDC.SMF22
                          DUMP.D1108.H20.#@$A.#MASTER#.S00014
                          DUMP.D1108.H20.#@$A.#MASTER#.S00015
                          DUMP.D1108.H20.#@$A.#MASTER#.S00017
                          DUMP.D1108.H21.#@$A.#MASTER#.S00022
ADR006I (001)-STEND(02), 2014.316 10:27:26 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.316 10:27:26 TASK COMPLETED WITH RETURN CODE 0000
PAGE 0002     5695-DF175  DFSMSDSS V2R01.0 DATA SET SERVICES     2014.316 10:27
ADR012I (SCH)-DSSU (01), 2014.316 10:27:26 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

The output of the job log shows that zEDC compressed data sets are completely transparent to the DFSMSdss **CONSOLIDATE** function. We then allocated a zEDC compressed data set on a storage management subsystem (SMS)-managed volume where non-compressed files were already allocated, and repeated the same test for this volume. The results were the same as expected, as shown in Example 6-3.

Example 6-3 Non-compressed files CONSOLIDATE

```
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR MARIO.SIA.MTAPLEX.IXLMGCSP.XML, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR IXGLOGR.CEA.000.CDFDDC0A.I9CE3445.A0000001.D, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR IXGLOGR.ADSW.CICSVR.F01DALDB.#@$2.DATA, 01
ADR260I (001)-DFRGD(01), EXTENTS REDUCED FROM 009 TO 001 FOR PBRES2.ZCOMP.BAM307.V1Q
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR KYNEF.AUTOEMCS.PANELS.XMIT, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR KYNEF.LLA.JCL, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR MARIO.XESCPUTS.SMFDATA, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR IXGLOGR.CEA.000.CCBFB9BA.N9B61160.A0000000.D, 01
ADR260I (001)-DFRGD(01), EXTENTS REDUCED FROM 002 TO 001 FOR DISTDB2.DSNDBD.DBCRWW1.TSTCK000.I0001.A008
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR DISTDB2.DSNDBD.DBCRWW1.XITEM000.I0001.A001, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR KYNEF.BEVERSO.SYSSTAT.R2.XMIT, 01
ADR261I (001)-DFRGD(01), UNABLE TO FURTHER CONSOLIDATE EXTENTS FOR KYNEF.BRODCAST, 01
```

Note that PBRES2.ZCOMP.BAM307.V1Q is a zEDC compressed file.

6.2.2 COPY

Data movement using **COPY** is necessary when you are performing the following tasks:

- Replacing devices

  When you remove devices to be replaced with other ones, you must move the data off the devices that you are removing.

- Adding devices

  If you add new devices at your site, you must move data onto them to use the added capacity.

- Maintaining devices

  When you are servicing a volume, you might need to move data off of the volume so that users can continue to access the data.

- Tuning performance

  If a volume is performing poorly, it might be because data sets on the volume are being frequently accessed and causing an I/O bottleneck. In this case, you might move the data sets to another volume that is better able to handle it (either because it is less full or because it is cached).

You can use the DFSMSdss **COPY** command to move data between volumes.

The DFSMSdss **COPY** command performs data set movement, volume movement, and track movement from one DASD volume to another.

You can copy data sets to another volume of either like or unlike device types. Like devices have the same track capacity (3390 Model 2 and 3390 Model 3), where unlike devices have different track capacities (3380 Model K and 3390 Model 3).

However, the DASD must be of *like* device type if you copy a full volume, range of tracks, or physically copy a data set. The user must specify the source volumes and the target volumes. DFSMSdss only allows one source and one target volume.

DFSMSdss offers two ways to process **COPY** commands:

- *Logical processing* is data set-oriented, which means that it operates against data sets and volumes independently of physical device format.
- *Physical processing* can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level. The processing method is determined by the keywords specified on the command.

> **Note:** The `REBLOCK` keyword is NOT supported on the **COPY** task for zEDC compressed format data sets.
>
> If used on z/OS V1R12 and V1R13, DFSMSdss will fail logical data set **COPY** operations of extended format data sets in the zEDC compressed format.

The goal of this test was to move a zEDC compressed data set from one location to another. We accomplished it by performing a physical and a logical DFSMSdss **COPY** operation. Figure 6-1 shows the characteristics of the source data set that we selected.

```
Data Set Information
Command ===>
                                                                  More:     +
Data Set Name . . . . : PBRES2.ZCOMP.BAM301.F2B

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 28
 Storage class  . . . : MANAGED         Allocated extents . : 1
  Volume serial . . . : TRC520
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS             Current Utilization
  Record format . . . : FB              Used cylinders  . . : 25
  Record length . . . : 80              Used extents  . . . : 1
  Block size  . . . . : 32720
  1st extent cylinders: 28
  Secondary cylinders : 5              Dates
  Data set name type  : EXTENDED        Creation date . . . : 2014/11/12
                                        Referenced date . . : 2014/11/15
                                        Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 6-1 Source data set for COPY

Example 6-4 shows a physical **COPY** operation of a zEDC compressed data set using DFSMSdss.

Example 6-4 Physical COPY of zEDC compressed data set

```
//JOB1        JOB ...............
//COPY        EXEC    PGM=ADRDSSU
//SYSPRINT    DD      SYSOUT=*
//SYSOUT      DD      SYSOUT=*
//SNAP        DD      SYSOUT=*
//DASD1       DD      UNIT=3390,VOL=(PRIVATE,SER=TRA351),DISP=SHR
//SYSIN       DD      *
```

```
   COPY DATASET(                            -
     INCLUDE(PBRES2.ZCOMP.BAM301.F2B)       -
     BY(MULTI,=,NO))                        -
     PHYSINDDNAME(DASD1)                    -
     OUTDYNAM(TRC120)                       -
     DELETE
/*
```

Figure 6-2 shows part of the job log outlining the results of the physical **COPY** operation.

```
  COPY DATASET(                            -
     INCLUDE(PBRES2.ZCOMP.BAM301.F2B)       -
     BY(MULTI,=,NO))                        -
     PHYSINDDNAME(DASD1)                    -
     OUTDYNAM(TRC120)                       -
     DELETE
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'COPY '
ADR109I (R/I)-RI01 (01), 2014.317 14:51:39 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2014.317 14:51:39 EXECUTION BEGINS
ADR396I (001)-PCNVS(01), DATA SET PBRES2.ZCOMP.BAM301.F2B ALLOCATED, ON VOLUME(S): TRC120
ADR431I (001)-DYNA (02), DATA SET PBRES2.ZCOMP.BAM301.F2B HAS BEEN DELETED
ADR465I (001)-PCNVX(01), DATA SET PBRES2.ZCOMP.BAM301.F2B HAS BEEN CATALOGED IN CATALOG UCAT.V#@$#M1
ADR801I (001)-DDDS (01), 2014.317 14:51:39 DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0
FAILED SERIALIZATION
                         AND 0 FAILED FOR OTHER REASONS
ADR454I (001)-DDDS (02), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          PBRES2.ZCOMP.BAM301.F2B
ADR006I (001)-STEND(02), 2014.317 14:51:39 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.317 14:51:39 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2014.317 14:51:39 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Figure 6-2 Results of COPY

Example 6-5 shows a logical **COPY** operation of a zEDC compressed data set using DFSMSdss.

Example 6-5 Logical COPY of zEDC compressed data set

```
//JOB1      JOB ...................
//COPY        EXEC    PGM=ADRDSSU
//SYSPRINT    DD      SYSOUT=*
//SYSOUT      DD      SYSOUT=*
//SNAP        DD      SYSOUT=*
//SYSIN       DD      *
   COPY DATASET(                            -
     INCLUDE(PBRES2.ZCOMP.BAM301.F2B)       -
     BY(MULTI,=,NO))                        -
     OUTDYNAM(TRA351)                       -
     DELETE
/*
```

Example 6-6 shows part of the job log outlining the results of the logical **COPY** operation.

Example 6-6 Results of the logical COPY

```
    COPY DATASET(                          -
      INCLUDE(PBRES2.ZCOMP.BAM301.F2B)     -
      BY(MULTI,=,NO))                      -
      OUTDYNAM(TRA351)                     -
      DELETE
 ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'COPY '
 ADR109I (R/I)-RI01 (01), 2014.319 10:44:21 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
 ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
0ADR006I (001)-STEND(01), 2014.319 10:44:21 EXECUTION BEGINS
0ADR711I (001)-NEWDS(01), DATA SET PBRES2.ZCOMP.BAM301.F2B HAS BEEN ALLOCATED USING STORCLAS MANAGED, DATACLAS
COMPZEDC, AND NO MGMTCLAS
0ADR806I (001)-TOMI (01), DATA SET PBRES2.ZCOMP.BAM301.F2B COPIED USING A FAST REPLICATION FUNCTION
0ADR431I (001)-CNVSM(02), DATA SET PBRES2.ZCOMP.BAM301.F2B IN CATALOG UCAT.V#@$#M1 HAS BEEN DELETED
0ADR801I (001)-DDDS (01), 2014.319 10:44:21 DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
SERIALIZATION
                          AND 0 FAILED FOR OTHER REASONS
0ADR454I (001)-DDDS (02), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
0                          PBRES2.ZCOMP.BAM301.F2B
0ADR006I (001)-STEND(02), 2014.319 10:44:21 EXECUTION ENDS
0ADR013I (001)-CLTSK(01), 2014.319 10:44:21 TASK COMPLETED WITH RETURN CODE 0000
0ADR012I (SCH)-DSSU (01), 2014.319 10:44:21 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

6.2.3 DEFRAG

Because of the nature of allocation algorithms and the frequent creation, extension, and deletion of data sets, free space on DASD volumes becomes fragmented. This can result in the following issues:

- Inefficient use of DASD storage space
- An increase in space-related abnormal end of tasks (abends)
- Performance degradation caused by excessive DASD arm movement
- An increase in the time required for functions that are related to direct access device space management (DADSM).

With the **DEFRAG** command, you can consolidate the free space on volumes and avoid these problems. The **DEFRAG** command relocates data set extents on a DASD volume to reduce or eliminate free space fragmentation, and prints a report about free space and other volume statistics. Also, you can specify which data sets, if any, are to be excluded from data-set-extent relocation. Data set extents are not combined as a result of **DEFRAG** processing.

Note: Data set extents are not moved between the track-managed space and cylinder-managed space of an extended address volume during **DEFRAG** processing.

Example 6-7 shows an example of a DFSMSdss **DEFRAG** operation.

Example 6-7 Sample of a DFSMSdss DEFRAG operation

```
//JOB       JOB ..............
//*
//DEFRAG       EXEC    PGM=ADRDSSU
//SYSPRINT     DD      SYSOUT=*
//SYSOUT       DD      SYSOUT=*
//SNAP         DD      SYSOUT=*
//DASD         DD      UNIT=3390,VOL=(PRIVATE,SER=#@$#Z2),DISP=OLD
//SYSIN        DD      *
 DEFRAG DDNAME(DASD) -
  EXCLUDE(LIST(PBRES2.ZEDC.DFDSS.DUMP1))
```

During this test, we selected the volume (#@$#Z2) previously assigned to the COMPZEDS Storage Group that was defined to hold zEDC compressed files exclusively.

Figure 6-3 shows the results of a **DEFRAG** operation. Although all of the files on this volume are zEDC compressed files, the DFSMSdss **DEFRAG** operation works with the same set of parameters, as with normal files.

```
 DEFRAG DDNAME(DASD) -
  EXCLUDE(LIST(PBRES2.ZEDC.DFDSS.DUMP1))
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DEFRAG '
ADR109I (R/I)-RI01 (01), 2014.317 16:49:27 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2014.317 16:49:27 EXECUTION BEGINS
ADR208I (001)-DFRGD(01), 2014.317 16:49:27 BEGINNING STATISTICS ON #@$#Z2:
                          FREE CYLINDERS                  00004481
                          FREE TRACKS                     00000004
                          FREE EXTENTS                    00000006
                          LARGEST FREE EXTENT (CYL,TRK) 00002312,00
                          FRAGMENTATION INDEX                0.088
                          PERCENT FREE SPACE                    44
ADR806I (001)-DFRGD(01), RELOCATED EXTENTS WILL BE COPIED USING A FAST REPLICATION FUNCTION
ADR213I (001)-DFANL(01), 2014.317 16:49:27 ENDING STATISTICS ON #@$#Z2:
                          DATA SET EXTENTS RELOCATED      00000002
                          TRACKS RELOCATED                00002340
                          FREE CYLINDERS                  00004481
                          FREE TRACKS                     00000004
                          FREE EXTENTS                    00000004
                          LARGEST FREE EXTENT (CYL,TRK) 00002300,00
                          FRAGMENTATION INDEX                0.071
ADR006I (001)-STEND(02), 2014.317 16:49:27 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.317 16:49:27 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2014.317 16:49:27 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Figure 6-3 Results of a DEFRAG operation

6.2.4 DUMP and RESTORE

You can use the DFSMSdss **DUMP** command to back up volumes and data sets, and you can use the DFSMSdss **RESTORE** command to recover them. You can make incremental backups of your data sets by specifying a data set **DUMP** command with **RESET**, and filtering on the data-set-changed indicator.

In an SMS environment, DFSMSdss saves the class names for the data sets that it dumps. When you restore the data set to an SMS-managed volume, DFSMSdss starts the automatic class selection (ACS) routines, and then passes the class names saved with the data set to them. Based on the class names and other input from DFSMSdss (for example, class names specified with the `STORCLAS` or `MGMTCLAS` keywords), ACS assigns SMS constructs to each data set.

Because DFSMSdss **RESTORE** starts ACS, you can restore the data sets to SMS-managed volumes. Conversely, data sets backed up as SMS-managed data sets can be restored as non-SMS-managed data sets.

In addition to providing for routine backup requirements, you can use DFSMSdss to back up application data for disaster recovery and vital records purposes. You can back up all of the data sets (including data that is only on the primary DASD, but you cannot use DFSMSdss to process migrated data sets) that are associated with a particular application for disaster recovery or vital records. You accomplish this backup by using DFSMSdss logical data set dump, and filtering on data set names.

If you do not want to perform a separate dump operation for disaster recovery, you can specify more than one `OUTDDNAME` to create up to 255 separate backup copies when you do your routine backup. These extra copies can then be used for disaster recovery or vital records purposes. The **DUMP** command can also be used to archive data sets that have not been accessed for long periods of time.

DFSMSdss can perform two kinds of processing when running **COPY**, **DUMP**, and **RESTORE** commands:

- *Logical processing* operates against data sets independently of physical device format.
- *Physical processing* moves data at the track-image level and operates against volumes, tracks, and data sets.

Each type of processing offers different capabilities and advantages.

During a restore operation, the data is processed the same way that it is dumped, because physical and logical dump tapes have different formats. If a data set is dumped logically, it is restored logically; if it is dumped physically, it is restored physically. A data set restore operation from a full-volume dump is a physical data set restore operation.

Note: If used on z/OS V1R12 and V1R13, DFSMSdss will fail logical data set **DUMP** operations of extended format data sets in the zEDC compressed format.

To explain the behavior of the DFSMSdss **DUMP** and **RESTORE** operations, we run several tests that we describe in the following order. We start with single file operations and then move on to full volume operations.

First, we run a **DUMP** and **RESTORE** operation on a single file.

The file we use as the source has the structure shown in Figure 6-4.

```
Data Set Information
Command ===>
                                                              More:     +
Data Set Name . . . . : PBRES2.ZCOMP.BAM309.OPTIMZQ

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 160
 Storage class  . . . : MANAGED         Allocated extents . : 1
  Volume serial . . . : TRC620
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS             Current Utilization
  Record format . . . : FSA             Used cylinders  . . : 158
  Record length . . . : 8906            Used extents  . . . : 1
  Block size  . . . . : 8906
  1st extent cylinders: 160
  Secondary cylinders : 10             Dates
  Data set name type  : EXTENDED        Creation date . . . : 2014/11/12
                                        Referenced date . . : 2014/11/13
                                        Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 6-4 Compressed source data set for DUMP and RESTORE

We run a DFSMSdss **DUMP** operation creating an uncompressed dump file, as shown in Example 6-8.

Example 6-8 DFSMSdss DUMP to create an uncompressed dump file

```
//JOB1      JOB .....................
//DUMPDSN      EXEC    PGM=ADRDSSU
//SYSPRINT     DD      SYSOUT=*
//SYSOUT       DD      SYSOUT=*
//SNAP         DD      SYSOUT=*
//OUTPUT       DD      DISP=(,CATLG,DELETE),
//             DSN=PBRES2.DFDSS.DUMP.OPTIMZQ,
//             SPACE=(CYL,(150,15),RLSE),
//             UNIT=3390
//SYSIN        DD      *
  DUMP DATASET(INCLUDE(                     -
        PBRES2.ZCOMP.BAM309.OPTIMZQ        ,-
              )) -
       OUTDD(OUTPUT)                        -
       OPTIMIZE(4)   TOL(ENQF)              -
       SHARE                                -
       SPHERE                               -
       CANCELERROR                          -
       ALLDATA(*)                           -
       ALLEXCP
//*
```

Example 6-9 shows an extract of the job log.

Example 6-9 Extract of DUMP job output for compressed dump file

```
DUMP DATASET(INCLUDE(                     -
        PBRES2.ZCOMP.BAM309.OPTIMZQ        ,-
              )) -
       OUTDD(OUTPUT)                        -
       OPTIMIZE(4)   TOL(ENQF)              -
       SHARE                                -
       SPHERE                               -
       CANCELERROR                          -
       ALLDATA(*)                           -
       ALLEXCP
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 2014.318 09:48:29 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2014.318 09:48:29 EXECUTION BEGINS
ADR903I (001)-PSEDM(01), DUMP OF EXTENDED SEQUENTIAL DATA SET PBRES2.ZCOMP.BAM309.OPTIMZQ WAS SUCCESSFUL. SIZE OF
DATA SET DUMPED
                         WAS 0000000011E3ECFC
ADR801I (001)-DTDSC(01), 2014.318 09:48:31 DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
SERIALIZATION
                         AND 0 FAILED FOR OTHER REASONS
ADR454I (001)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          PBRES2.ZCOMP.BAM309.OPTIMZQ
ADR006I (001)-STEND(02), 2014.318 09:48:31 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.318 09:48:31 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2014.318 09:48:31 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Because the source file, `PBRES2.ZCOMP.BAM309.OPTIMZQ`, is an extended-format sequential data set that was compressed using zEDC services, DFSMSdss inserts the information about the size of the extended-sequential data set.

Figure 6-5 shows the characteristics of the resulting dump data set.

```
Data Set Information
Command ===>
                                                                   More:     +
Data Set Name . . . . : PBRES2.DFDSS.DUMP.OPTIMZQ

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 237
 Storage class  . . . : **None**        Allocated extents . : 8
  Volume serial . . . : WORKW3
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : U               Used cylinders  . . : 237
  Record length . . . : 0               Used extents  . . . : 8
  Block size  . . . . : 27998
  1st extent cylinders: 30
  Secondary cylinders : 15             Dates
  Data set name type  :                 Creation date . . . : 2014/11/14
                                        Referenced date . . : 2014/11/14
                                        Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 6-5 Resulting dump data set

Note that the dump data set is *not* zEDC compressed. In the next step, we reproduce the **DUMP**, except that we create a dump data set that is zEDC compressed. Example 6-10 shows the sample JCL.

Example 6-10 Compressed dump of data set zEDC compressed

```
//JOB1      JOB ..................
//DUMPDSN      EXEC    PGM=ADRDSSU PARM='TYPRUN=NORUN'
//SYSPRINT     DD      SYSOUT=*
//SYSOUT       DD      SYSOUT=*
//SNAP         DD      SYSOUT=*
//OUTPUT       DD      DISP=(,CATLG,DELETE),
//             DSN=PBRES2.ZEDC.DFDSS.DUMP.OPTIMZQ,
//             SPACE=(CYL,(150,15),RLSE),
//             DATACLAS=COMPZEDC,
//             UNIT=3390
//SYSIN        DD       *
  DUMP DATASET(INCLUDE(                       -
        PBRES2.ZCOMP.BAM309.OPTIMZQ          ,-
              )) -
       OUTDD(OUTPUT)                          -
       OPTIMIZE(4)   TOL(ENQF)                -
       SHARE                                  -
       SPHERE                                 -
       CANCELERROR                            -
       ALLDATA(*)                             -
       ALLEXCP
//*
```

Note that we have changed the name of the dump data set, and inserted another jobcard specifying the correct data class to start zEDC compression services. Example 6-11 shows an extract of the JES2 job log indicating that the dump data set will get zEDC compression services.

Example 6-11 Dump data set gets zEDC compression services

```
IGD17070I DATA SET PBRES2.ZEDC.DFDSS.DUMP.OPTIMZQ
ALLOCATED SUCCESSFULLY WITH 1 STRIPE(S).
IGD17160I DATA SET PBRES2.ZEDC.DFDSS.DUMP.OPTIMZQ
IS ELIGIBLE FOR COMPRESSION
IGD101I SMS ALLOCATED TO DDNAME (OUTPUT  )
        DSN (PBRES2.ZEDC.DFDSS.DUMP.OPTIMZQ              )
        STORCLAS (MANAGED) MGMTCLAS (        ) DATACLAS (COMPZEDC)
        VOL SER NOS= TRD15E
```

Example 6-12 shows the resulting DFSMSdss log for this **DUMP** operation.

Example 6-12 Resulting DFSMSdss log

```
DUMP DATASET(INCLUDE(                     -
        PBRES2.ZCOMP.BAM309.OPTIMZQ        ,-
              )) -
       OUTDD(OUTPUT)                        -
       OPTIMIZE(4)   TOL(ENQF)              -
       SHARE                                -
       SPHERE                               -
       CANCELERROR                          -
       ALLDATA(*)                           -
       ALLEXCP
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 2014.318 10:41:46 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2014.318 10:41:46 EXECUTION BEGINS
ADR903I (001)-PSEDM(01), DUMP OF EXTENDED SEQUENTIAL DATA SET PBRES2.ZCOMP.BAM309.OPTIMZQ WAS SUCCESSFUL. SIZE OF
DATA SET DUMPED
                         WAS 0000000011E3ECFC
ADR801I (001)-DTDSC(01), 2014.318 10:41:47 DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
SERIALIZATION
                         AND 0 FAILED FOR OTHER REASONS
ADR454I (001)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          PBRES2.ZCOMP.BAM309.OPTIMZQ
ADR006I (001)-STEND(02), 2014.318 10:41:47 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.318 10:41:47 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2014.318 10:41:47 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Figure 6-6 shows the data set characteristics of the zEDC compressed dump data set.

```
Data Set Information
Command ===>
                                                                  More:     +
Data Set Name . . . . : PBRES2.ZEDC.DFDSS.DUMP.OPTIMZQ

General Data                       Current Allocation
 Management class . . : **None**    Allocated cylinders : 42
 Storage class  . . . : MANAGED     Allocated extents . : 1
  Volume serial . . . : TRD15E
  Device type . . . . : 3390
 Data class . . . . . : COMPZEDC
  Organization  . . . : PS         Current Utilization
  Record format . . . : U           Used cylinders  . . : 42
  Record length . . . : 80          Used extents  . . . : 1
  Block size  . . . . : 32720
  1st extent cylinders: 42
  Secondary cylinders : 15         Dates
  Data set name type  : EXTENDED    Creation date . . . : 2014/11/14
                                    Referenced date . . : 2014/11/14
                                    Expiration date . . : ***None***
  SMS Compressible  . : YES
```

Figure 6-6 Data set characteristics of the zEDC compressed dump data set

Notice that in this example we have determined that the characteristics of the resulting dump data set. By coding the appropriate **DATACLAS** parameter in job control language (JCL), we do not influence the content of the dump data set. Remember that it is not possible to write an extended-format sequential data set to tape.

By adding support for zEDC in DFSMSdss, it is possible to cause the zEDC compression to take place during the actual **DUMP** operation, therefore compacting the contents of the dump file at execution time. Example 6-13 shows an example JCL.

Example 6-13 JCL for support for zEDC in DFSMSdss

```
//JOB1      JOB ...............
//DUMPDSN      EXEC    PGM=ADRDSSU
//SYSPRINT     DD      SYSOUT=*
//SYSOUT       DD      SYSOUT=*
//SNAP         DD      SYSOUT=*
//OUTPUT       DD      DISP=(,CATLG,DELETE),
//             DSN=PBRES2.DFDSS.DUMP.OPTIMZQ,
//             SPACE=(CYL,(150,15),RLSE),
//             UNIT=3390
//SYSIN        DD       *
  DUMP DATASET(INCLUDE(                     -
        PBRES2.ZCOMP.BAM309.OPTIMZQ        ,-
              )) -
       OUTDD(OUTPUT)                        -
       OPTIMIZE(4)   TOL(ENQF)              -
       SHARE                                -
       SPHERE                               -
       CANCELERROR                          -
```

```
      ALLDATA(*)                              -
      ZCOMPRESS(PREF)                         -
      ALLEXCP
//*
```

Note that we don't call the zEDC compression services at the JCL level, but in SYSIN. The dump data set characteristics slightly differ from those produced before. See Figure 6-7.

```
Data Set Information
Command ===>
                                                                    More:     +
Data Set Name . . . . : PBRES2.DFDSS.DUMP.OPTIMZQ

General Data                           Current Allocation
 Management class . . : **None**        Allocated cylinders : 73
 Storage class  . . . : **None**        Allocated extents . : 1
  Volume serial . . . : WORKW3
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS             Current Utilization
  Record format . . . : U               Used cylinders  . . : 73
  Record length . . . : 0               Used extents  . . . : 1
  Block size  . . . . : 27998
  1st extent cylinders: 73
  Secondary cylinders : 15             Dates
  Data set name type  :                 Creation date . . . : 2014/11/18
                                        Referenced date . . : 2014/11/18
                                        Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 6-7 Dump data set characteristics

Note that the dump data set is *not* extended-format zEDC compressed, but still significantly smaller then the uncompressed dump data set (see Table 6-1). By using the **ZCOMPRESS(PREF)** parameter and not compressing the resulting output data set, this file is also eligible to be written on tape.

If the contents of the dump data set is compressed using zEDC **ZCOMPRESS(PREF)** and not the output file, transmitting the file to other locations also benefits from the much smaller size.

Table 6-1 shows the different file sizes according to the different usage of the zEDC compression services.

Table 6-1 File sizes by compression type

Source data set size (Cyl) zEDC compressed	Target data set size (Cyl) uncompressed	Target data set size zEDC compressed (Cyl)	Target data set size using SYSIN ZCOMPRESS(PREF)
158	237	42	73

As a next step we did a full volume dump. The source volume has a total of 30,051 cylinders available and is 83% (25,543 cylinders) used. Table 6-2 describes the contents of the source 3390 Model 27 volume.

Table 6-2 Full volume dump: Source volume

DSORG	# of files	# of tracks	% of space used on volume
Hierarchical file system (HFS)	3	2,002	0.44
Partitioned organization (PO)	391	26,015	5.77
Partitioned organization extended (PO-E)	46	5,634	1.25
Physically sequential (PS)	1,192	255,594	56.70
VSAM (VS)	45	89,181	19.78

Example 6-14 shows the sample JCL used to produce the non-extended-format sequential dump data set.

Example 6-14 JCL for non-extended-format sequential dump data set

```
//JOB1      JOB ............
//DUMPDSN      EXEC    PGM=ADRDSSU
//SYSPRINT     DD      SYSOUT=*
//SYSOUT       DD      SYSOUT=*
//SNAP         DD      SYSOUT=*
//INPUT        DD      DISP=SHR,
//             UNIT=3390,VOL=SER=BOX001
//OUTPUT       DD      DISP=(,CATLG,DELETE),
//             DSN=PBRES2.ZEDC.DFDSS.DUMP.BOX0013,
//             SPACE=(CYL,(3900,3900),RLSE),
//             DSNTYPE=LARGE,
//             UNIT=(3390)
//SYSIN        DD        *
  DUMP  -
  INDD(INPUT) -
  OUTDDNAME(OUTPUT) -
  ALLEXCP  -
  ALLDATA(*) -
  ZCOMPRESS(PREF) -
  OPTIMIZE(4)
```

To ensure that the output dump data set is not compressed by zEDC, we did not specify any `DATACLAS` jobcard. Because of the missing data class indication, and because the ACS routines in our example will not enforce a `DATACLASS` attribute for our data set pattern, we made sure that no compression took place for the dump data set.

By specifying the DFSMSdss **`ZCOMPRESS(PREF)`** parameter, we defined that the contents of the dump data set are compressed using zEDC compression services. This enables an increased throughput and a smaller dump data set that can also be written to tape.

Example 6-15 shows part of the job log, including the message confirming that zEDC has been used to compress the contents of the dump data set.

Example 6-15 Joblog confirming zEDC use

```
DUMP  -
  INDD(INPUT) -
  OUTDDNAME(OUTPUT) -
  ALLEXCP  -
  ALLDATA(*) -
  COMPRESS -
  ZCOMPRESS(PREF) -
  OPTIMIZE(4)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 2014.322 14:30:44 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR533I (001)-ZCOMP(01), 2014.322 14:30:44 ZEDC SERVICES TO BE USED FOR DUMP DATA SET ON DDNAME OUTPUT
ADR006I (001)-STEND(01), 2014.322 14:30:44 EXECUTION BEGINS
ADR006I (001)-STEND(02), 2014.322 14:33:02 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.322 14:33:02 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2014.322 14:33:02 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Also note that the ADR533I message is displayed in the JES2 job message log of the batch job.

Figure 6-8 shows the data set characteristics of the dump data set. Note that the size of the data set is reduced by a ratio of 3:1 when compared with the source volume.

```
Data Set Information
Command ===>
                                                                  More:     +
Data Set Name . . . . : PBRES2.ZEDC.DFDSS.DUMP.BOX0013

General Data                          Current Allocation
 Management class . . : MCDB22         Allocated cylinders : 7,662
 Storage class  . . . : STORAGE        Allocated extents . : 2
  Volume serial . . . : SBOX3G
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS            Current Utilization
  Record format . . . : U              Used cylinders  . . : 7,662
  Record length . . . : 0              Used extents  . . . : 2
  Block size  . . . . : 27998
  1st extent cylinders: 3900
  Secondary cylinders : 3900          Dates
  Data set name type  : LARGE          Creation date . . . : 2014/11/18
                                       Referenced date . . : 2014/11/18
                                       Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 6-8 Data set characteristics of the dump data set

To compare the difference in size of the DFSMSdss dump data sets, we did the same **DUMP** again, but not using any compression at all. Example 6-16 on page 103 shows the JCL used for the full volume dump without any compression.

Example 6-16 JCL for the volume dump without compression

```
//JOB1      JOB ...........
//DUMPDSN      EXEC    PGM=ADRDSSU
//SYSPRINT     DD      SYSOUT=*
//SYSOUT       DD      SYSOUT=*
//SNAP         DD      SYSOUT=*
//INPUT        DD      DISP=SHR,
//             UNIT=3390,VOL=SER=BOX001
//OUTPUT       DD      DISP=(,CATLG,DELETE),
//             DSN=PBRES2.ZEDC.DFDSS.DUMP.BOX0010,
//             SPACE=(CYL,(3900,3900),RLSE),
//             DSNTYPE=LARGE,
//             UNIT=(3390)
//SYSIN        DD       *
  DUMP  -
  INDD(INPUT) -
  OUTDDNAME(OUTPUT) -
  ALLEXCP  -
  ALLDATA(*) -
  OPTIMIZE(4)
```

Figure 6-9 shows the characteristics of the uncompressed DFSMSdss dump file.

```
Data Set Information
Command ===>
                                                                More:     +
Data Set Name . . . . : PBRES2.ZEDC.DFDSS.DUMP.BOX0010

General Data                      Current Allocation
 Management class . . : MCDB22     Allocated cylinders : 23,165
 Storage class  . . . : STORAGE    Allocated extents . : 6
  Volume serial . . . : SBOX3H
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS        Current Utilization
  Record format . . . : U          Used cylinders  . . : 23,165
  Record length . . . : 0          Used extents  . . . : 6
  Block size  . . . . : 27998
  1st extent cylinders: 3900
  Secondary cylinders : 3900      Dates
  Data set name type  : LARGE      Creation date . . . : 2014/11/18
                                   Referenced date . . : 2014/11/18
                                   Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 6-9 Characteristics of the zEDC uncompressed DFSMSdss dump file

Note that the size of the dump data set is now almost the same as the total of the allocated space on the volume. We also recognized a drop in elapsed time when we compared the two batch jobs. For the job that produced the uncompressed dump data set, elapsed time was 3 min. 36 sec. The elapsed time for the batch job that produced the zEDC compressed dump data set was 2 min. 18 sec.

The next step is the execution of a **DUMP** and **RESTORE** operation, and a check for data consistency. For this test we selected an Extensible Markup Language (XML) file as input. Figure 6-10 shows the characteristics of the data set.

```
Data Set Information
Command ===>
                                                                   More:     +
Data Set Name . . . . : PBRES2.NOCOMP.NOREC.BAM302.XMLETM

General Data                          Current Allocation
 Management class . . : **None**       Allocated cylinders : 209
 Storage class  . . . : **None**       Allocated extents . : 20
  Volume serial . . . : WORKW3 +
  Device type . . . . : 3390
 Data class . . . . . : **None**
  Organization  . . . : PS            Current Utilization
  Record format . . . : VB             Used cylinders  . . : 209
  Record length . . . : 4096           Used extents  . . . : 20
  Block size  . . . . : 27998
  1st extent cylinders: 20
  Secondary cylinders : 10            Dates
  Data set name type  :                Creation date . . . : 2014/11/06
                                       Referenced date . . : 2014/11/15
                                       Expiration date . . : ***None***
  SMS Compressible  . : NO
```

Figure 6-10   XML file characteristics

Example 6-17 shows an extract of the contents.

Example 6-17   XML file contents

```
<FieldValue>--rit=3
--lpar=1
--machine=2097
--noscan
--nobytes
--timeData
--opcode
--enableThread
--noModuleDetails
--swreport
/tmp/zconv.STJOHN.IBMBASIC.L1019TST/IBMBASIC.L1019TST.SWREPORT
/tmp/zconv.STJOHN.IBMBASIC.L1019TST/IBMBASIC.L1019TST.IAREDUCE
/tmp/zconv.STJOHN.IBMBASIC.L1019TST/IBMBASIC.L1019TST.MAPZOS
/tmp/zconv.STJOHN.IBMBASIC.L1019TST/IBMBASIC.L1019TST.ETM</FieldValue>
</Detail>
<Detail>
<FieldTitle>Converter</FieldTitle>
<FieldValue>TraceReader Version 1.8.4 (ADTools V. 03 Nov 2010)</FieldValue>
</Detail>
<Detail>
```

We ran a DFSMSdss logical **DUMP** operation with two steps. Step one produced a non-compressed dump data set, and step two produced a zEDC compressed dump data set.

Then we ran a DFSMSdss **RESTORE** operation with two steps. Step one restored the data set from the non-compressed dump data set and renamed it to a new name, and step two did the same with the zEDC compressed dump data set. See Example 6-18 for the **RESTORE** operation.

Example 6-18 RESTORE of zEDC compressed dump data set

```
//JOB1      JOB .................
//RSTNZ        EXEC     PGM=ADRDSSU
//*----------------------------------------------------------------------
//SYSPRINT     DD       SYSOUT=*
//IN01         DD       DISP=SHR,
//             DSN=PBRES2.ZEDC.DFDSS.DUMP.XMLETM.NZEDC
//SYSIN        DD       *
 RESTORE DATASET(INCLUDE(*.**)) -
   INDD(IN01)                   -
   SPHERE                       -
   RENAMEU(                     -
   (PBRES2.NOCOMP.NOREC.BAM302.XMLETM,        -
    PBRES2.NOCOMP.NOREC.BAM302.XMLETM.RSTNZ)) -
   RECATALOG(*)
//*
//RSTZ         EXEC     PGM=ADRDSSU
//*----------------------------------------------------------------------
//SYSPRINT     DD       SYSOUT=*
//IN01         DD       DISP=SHR,
//             DSN=PBRES2.ZEDC.DFDSS.DUMP.XMLETM.ZEDC
//SYSIN        DD       *
 RESTORE DATASET(INCLUDE(*.**)) -
   INDD(IN01)                   -
   SPHERE                       -
   RENAMEU(                     -
   (PBRES2.NOCOMP.NOREC.BAM302.XMLETM,       -
    PBRES2.NOCOMP.NOREC.BAM302.XMLETM.RSTZ)) -
   RECATALOG(*)
/*
```

See Example 6-19 for the results of the **COMPARE** operation to confirm data consistency.

Example 6-19 Output of COMPARE operation

```
NEW: PBRES2.NOCOMP.NOREC.BAM302.XMLETM.RSTNZ              OLD: PBRES2.NOCOMP.NOREC.BAM302.XMLETM

                        LINE COMPARE SUMMARY AND STATISTICS


  4162692 NUMBER OF LINE MATCHES               0  TOTAL CHANGES (PAIRED+NONPAIRED CHNG)
        0 REFORMATTED LINES                    0  PAIRED CHANGES (REFM+PAIRED INS/DEL)
        0 NEW FILE LINE INSERTIONS             0  NON-PAIRED INSERTS
        0 OLD FILE LINE DELETIONS              0  NON-PAIRED DELETES
  4162692 NEW FILE LINES PROCESSED
  4162692 OLD FILE LINES PROCESSED

 LISTING-TYPE = DELTA      COMPARE-COLUMNS =    1:4092      LONGEST-LINE = 354
 PROCESS OPTIONS USED: NONE

 ISRS004I LISTING LINES MAY BE TRUNCATED DUE TO LIMITING OUTPUT LINE WIDTH.
```

```
1  ISRSUPC   -   MVS/PDF FILE/LINE/WORD/BYTE/SFOR COMPARE UTILITY- ISPF FOR z/OS        2014/11/15  12.25    PAGE
1
 NEW: PBRES2.NOCOMP.NOREC.BAM302.XMLETM.RSTZ                  OLD: PBRES2.NOCOMP.NOREC.BAM302.XMLETM

                        LINE COMPARE SUMMARY AND STATISTICS



  4162692 NUMBER OF LINE MATCHES              0  TOTAL CHANGES (PAIRED+NONPAIRED CHNG)
        0 REFORMATTED LINES                   0  PAIRED CHANGES (REFM+PAIRED INS/DEL)
        0 NEW FILE LINE INSERTIONS            0  NON-PAIRED INSERTS
        0 OLD FILE LINE DELETIONS             0  NON-PAIRED DELETES
  4162692 NEW FILE LINES PROCESSED
  4162692 OLD FILE LINES PROCESSED

 LISTING-TYPE = DELTA      COMPARE-COLUMNS =    1:4092      LONGEST-LINE = 354
 PROCESS OPTIONS USED: NONE

 ISRS004I LISTING LINES MAY BE TRUNCATED DUE TO LIMITING OUTPUT LINE WIDTH.
```

6.2.5 PRINT

With the **PRINT** command, you can print the following information:

- A single-volume non-Virtual Storage Access Method (VSAM) data set, as specified by a fully qualified name. You must specify the volume where the data set is, but you do not need to specify the range of tracks it occupies.
- A single-volume VSAM data set component (not cluster). The component name specified must be the name in the volume table of contents (VTOC), not the name in the catalog.
- Ranges of tracks.
- All or part of the VTOC. The VTOC location does not need to be known.

Example 6-20 shows an example of a DFSMSdss **PRINT** operation on the zEDC-enabled data set.

Example 6-20 Sample of DFSMSdss PRINT operation

```
//JOB1        JOB .................
//PRINT       EXEC     PGM=ADRDSSU
//SYSPRINT    DD       SYSOUT=*
//SYSIN       DD       *
  PRINT DATASET(PBRES2.ZCOMP.BAM302.V1B) INDYNAM(TRA755)
/*
```

Figure 6-11 shows an extract of the output of the **PRINT** command.

```
 PRINT DATASET(PBRES2.ZCOMP.BAM302.V1B) INDYNAM(TRA755)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'PRINT '
ADR109I (R/I)-RI01 (01), 2014.317 17:54:49 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 2014.317 17:54:49 EXECUTION BEGINS
*** TRACK(CCHH)  01110000         R0 DATA  0000000000000000
    COUNT  011100000100DD58
 0000  460003FA 41000008 0002B0F7 2D007616  D5FD73F7 EE239B04 B2798777 80000142
*...?.......7....N..7......g.....*
 0020  082109F1 7DF10168 44A2A246 450DBE4A  1B1FB1B5 15298F20 8487F511 E5ED3302
*...1'1...ss....¢........dg5.V...*
 0040  5A6AAD0D 955A6A6D 619FC95A 6B43F695  B5D6D2BB 9B87D6FA BF77EFDD DDBBD6B2
*!¦..n!¦_/.I!,.6n.OK..gO?......O.*
 0060  FFDF9999 DF77BFE7 BDDF7777 17D402D9  FBCD7CDF CC3973CE 9C393373 E6CC8C52
*..rr...X.....M.R..@.........W...*
 ....
 DD00  EFDA79F7 2D9FF7AD 72000000 00000000  00000000 00000000 00000000 00000000
*...7..7.........................*
 DD20  00000000 00000000 00000000 0000000C  88000873 14FBC833 DD380000 00000000
*................h.....H.........*
 DD40  000D1000 00000000 00000051 00000000  00000000 005A5AA5                    *.....................!!v
*
ADR006I (001)-STEND(02), 2014.317 17:54:50 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 2014.317 17:54:50 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 2014.317 17:54:50 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Figure 6-11   Extract of the output of the PRINT command

7

zEDC and DFSMShsm when using DFSMSdss as the data mover

In this chapter, we describe how to use IBM zEnterprise Data Compression (zEDC) compression services in the Data Facility Storage Management Subsystem hierarchical storage manager (DFSMShsm) environment. We give you a short introduction of DFSMShsm, and then provide explanations of the DFSMShsm functions that use zEDC compression services. We list the necessary prerequisites, in addition to the parameters necessary to enable the zEDC function.

The chapter contains the following topics:

- Introduction
- Compression in DFSMShsm today
- Using zEDC support in DFSMShsm

© Copyright IBM Corp. 2015. All rights reserved.

7.1 Introduction

DFSMShsm is a functional component of the DFSMS family, which provides facilities for managing your storage devices. The following list describes the five components of DFSMS and their functions:

- DFSMS data set services (DFSMSdfp)

  Provides storage, data, program, and device management functions through the storage management subsystem (SMS).

- DFSMS removable media manager (DFSMSrmm)

  Provides tape management functions for removable media, such as virtual and physical tape cartridges.

- DFSMS data set services (DFSMSdss)

  Provides data movement, copy, backup, and space management functions in a batch environment.

- DFSMS object access method (DFSMSoam)

  Provides tape hardware management, such as cartridge entry, eject, and tape configuration database (TCDB) management.

- DFSMShsm

  Provides backup, recovery, migration, and space management functions with optimum automation capabilities.

7.1.1 DFSMShsm

DFSMShsm provides space, availability, and tape mount management functions in a storage device hierarchy for both system-managed, and non-system-managed storage environments. DFSMShsm enables you to automate your storage management tasks, improving the productivity by effectively managing the storage devices.

DFSMShsm cooperates with the other products in the DFSMS family to provide efficient and effective storage management.

The storage management subsystem provides storage groups, storage classes, management classes, and data classes that control the allocation parameters and management attributes of data sets. DFSMShsm performs space management and availability management of each data set as directed by the management class attributes of that data set.

DFSMShsm categorizes storage devices into three levels of storage devices. The hierarchy is used in its automatic management of data, relieving users from manual storage management tasks. In addition, the storage group controls the allocation of the data set when DFSMShsm returns the data set to level 0 (L^0) storage, the level where data sets that are directly accessible to the jobs that you run.

7.2 Compression in DFSMShsm today

DFSMShsm uses either DFSMSdss **COMPRESS** or **HWCOMPRESS** to compress user data during full-volume dump.

DFSMShsm uses its own *host-based compression algorithm* to compress user data. If you specify the compaction during migration option, DFSMShsm compacts each data set as it migrates. The first time the data set migrates, DFSMShsm always compacts it. If a migrated data set is recalled and is again a candidate for migration, DFSMShsm checks the compaction history in the migration control data set (MCDS) for that data set.

If the compaction from the earlier migration did not result in saving at least the percentage of space that you specified with the **COMPACTPERCENT** parameter of the **SETSYS** command, DFSMShsm does not perform compaction again.

> **Important:** If the data set is a Storage Access Method (SAM) or Virtual SAM (VSAM) compressed data set, DFSMShsm suspends compaction during migration, and then records in the MCDS that the data set is striped, compressed, or both.

If the data set is to migrate to an small-data-set packing (SDSP), the compaction size is always estimated using an internal default of 50%. Previous compaction history is not considered.

7.2.1 Specifying when compression with zEDC should be done

ZCOMPRESS is an optional parameter set that you use to specify the type of compression used during migration or backup for all data sets. The optional subparameters of the **ZCOMPRESS** parameter follow:

- `ALL | NONE`
- `DASDBACKUP (YES | NO)`
- `DASDMIGRATE (YES | NO)`
- `TAPEBACKUP (YES | NO)`
- `TAPEMIGRATE (YES | NO)`

> **Note:** The **ZCOMPRESS** parameter is used only for SMS data sets that are not compressed on L^0. Non-SMS data sets cannot be compressed on L^0.

> **SETSYS default:** If you do not specify the **ZCOMPRESS** parameter, the **SETSYS** command default is *no compression with zEDC*.
>
> **DFSMShsm default:** If you do not specify a subparameter with this parameter on any **SETSYS** command, the DFSMShsm default is *no compression with zEDC*.

For details, see *DFSMShsm Storage Administration Guide*, SC23-6871.

Example 7-1 shows an extract of the ARCMDxx member of SYS1.PARMLIB on our system.

Example 7-1 Extract of the ARCMDxx member

```
/****************************************************************/
/*               DFSMSHSM COMPACTION OPTIONS                    */
/****************************************************************/
/*                                                              */

SETSYS                      /* COMPACT DATA SETS THAT MIGRATE TO */ -
  COMPACT(DASDMIGRATE)      /* DASD.                             */

SETSYS                      /* DO NOT COMPACT DATA UNLESS A      */ -
    COMPACTPERCENT(20)      /* SAVINGS OF 20% OR MORE CAN BE     */
                            /* GAINED.                           */

/****************************************************************/
/* DFHSM ZCOMPRESSION OPTIONS                                   */
/* ADDITIONALLY ZOMPRESS( YES NO ) CAN BE SPECIFIED ON DUMPCLASS */
/****************************************************************/
   SETSYS ZCOMPRESS(DASDBACKUP(YES))
   SETSYS ZCOMPRESS(DASDMIGRATE(YES))
   SETSYS ZCOMPRESS(TAPEBACKUP(YES))
   SETSYS ZCOMPRESS(TAPEMIGRATE(YES))
```

DFSMShsm does use DFSMSdss **COMPRESS** or **HWCOMPRESS** to compress user data during full-volume dump.

DFSMShsm uses the DFSMSdss zEDC support in the following actions:

- **MIGRATE** / **RECALL**
- **BACKUP** / **RECOVER**
- **FULL VOLUME DUMP**
- **FRBACKUP DUMP**
- **RECOVER from DUMP** and **FRRECOV from DUMP**

DFSMShsm calls DFSMSdss with the **ZCOMPRESS(PREFERRED)** option.

COMPACTPERCENT works with **ZCOMPRESS** as it does for **COMPACT**. **COMPACTPERCENT** is an optional parameter specifying the percentage of space saved if DFSMShsm compacts all data sets. For **COMPACTPERCENT(pct)**, substitute a decimal number 0 - 99 to specify the least percentage amount of space you want saved if DFSMShsm compacts a data set.

If you request compaction, DFSMShsm compacts a data set when it migrates or backs up the data set for the first time. DFSMShsm then compares the number of bytes written to the total bytes of the original data set, and computes the percentage of bytes saved. If the percentage saved is not greater than or equal to the value defined in **COMPACTPERCENT(pct)**, DFSMShsm does not compact the data set during subsequent migrations or backups.

DFSMShsm does not check whether the data set was compacted during migration if DFSMShsm is currently backing up the data set. Similarly, DFSMShsm does not check whether the data set was compacted during backup if DFSMShsm is currently migrating the data set.

If zEDC services are not available at the time of the backup or migration, DFSMShsm looks at the values specified in the **COMPACT SETSYS** parameter. See Table 7-1 for the possible combinations.

Table 7-1 DFSMShsm compression alternatives

ZCOMPRESS	COMPACT	Results
None	None	DFSMShsm creates a backup or migrates the data set without using any form of compression.
None	All	DFSMShsm creates a backup or migrates the data set by using its current form of compression.
All	None	DFSMShsm attempts to use zEDC services to compress backup. If the services are unavailable, backup is uncompressed.
All	All	DFSMShsm attempts to use zEDC services to compress backup. If the services are unavailable, backup is compressed using its current form of compression.

7.3 Using zEDC support in DFSMShsm

The **SETSYS** command and its parameters are used to establish a DFSMShsm environment. When DFSMShsm is installed, a default set of **SETSYS** parameters is used. You can specify one or more **SETSYS** commands in the ARCCMDxx PARMLIB member that is used during the startup of DFSMShsm, or you can issue **SETSYS** commands with specific parameter values after DFSMShsm is started.

Alternatively, if you now specify zEDC compression rather than compaction (using the **ZCOMPRESS** parameter of the **SETSYS** command), zEDC compression takes place rather than compaction. If zEDC services are not available, DFSMShsm uses the **SETSYS COMPACT** settings to determine what type of software compaction is used for migration.

See "ZCOMPRESS: Specifying when compression with zEDC should be done" in *DFSMShsm Storage Administration Guide*, SC23-6871.

7.3.1 Overriding SETSYS for individual data sets

You can use DFSMShsm installation exits to customize DFSMShsm processing.

The DFSMShsm installation exits fall into two categories: Exits that support basic DFSMShsm functions, and exits that support DFSMShsm aggregate backup and recovery support (ABARS) functions. This section describes only the exits that support the basic DFSMShsm basic functions and the zEDC Express feature. They are listed in Table 7-2.

Table 7-2 DFSMShsm Installation Exits in support of zEDC Express feature

Module name	Description	When available
ARCBDEXT	Data set backup exit	During volume backup, when a data set fulfills the selection criteria. Also during command backup of individual data sets.
ARCMDEXT	Space management exit	When a data set fulfills the selection criteria for the level 0 volume being managed, but before the data set migrates or transitions.

The ARCBDEXT installation exit, called during volume backup processing, receives control after DFSMShsm determines that a data set should be backed up but before DFSMShsm backs it up. It also receives control during the backup of individual data sets through the **BACKDS** and **HBACKDS** commands, and during the backup of migrated data sets. ARCBDEXT can bypass compression for a particular data set when the following properties are set:

- **SETSYS(ZCOMPRESS(ALL))**
- **SETSYS(ZCOMPRESS(DASDBACKUP(YES)))**
- **SETSYS(ZCOMPRESS(TAPEBACKUP(YES)))**

You can use the data set backup exit (ARCBDEXT) to perform the following tasks:

- To prevent DFSMShsm from backing up selected data sets whenever volume backup processes the level 0 volumes on which the data sets are.
- To exclude non-SMS-managed data sets from backup as an alternative to using the **ALTERDS** command. This technique is effective for excluding large numbers of non-system-managed data sets from backup. For example, you can design the exit to make decisions based on data in the data set VTOC entry by selecting data sets based on part of the data set qualifier.
- To prevent compaction of a data set during volume backup to tape, DASD, or both, whenever you have previously specified one of the following commands:
  - **SETSYS COMPACT(TAPEBACKUP)**
  - **SETSYS COMPACT(DASDBACKUP)**
  - **SETSYS COMPACT(ALL)**
  - **SETSYS ZCOMPRESS(TAPEBACKUP(YES))**
  - **SETSYS ZCOMPRESS(DASDBACKUP(YES))**
  - **SETSYS ZCOMPRESS(ALL)**
- To direct DFSMShsm whether serialization should, or should not, be attempted before backing up the current data set, and whether a backup should be performed if serialization has been attempted but fails.
- To specify a RETAINDAYS value for the backup of a given data set.

> **Tip:** Do *not* use the ARCBDEXT installation exit to override management class parameters for a data set. However, you can use it to change the compaction rules for system-managed data sets.

The ARCMDEXT installation exit receives control whenever a data set fulfills the selection criteria for the level 0 volume being managed, but before the data set migrates or transitions. It is called when DFSMShsm processes a level 0 volume, or an individual data set, through any of the following ways:

- **HMIGRATE** command
- **MIGRATE** command
- Automatic primary space management
- Interval migration
- On-demand migration
- Class transitions

The input data structure provides flags that identify the type of volume migration function under which the exit was started.

You can use the space management exit (ARCMDEXT) to perform, among others, the prevention of compaction of a data set during volume migration if you have previously specified one of the following properties:

- **SETSYS COMPACT(TAPEMIGRATE)**
- **SETSYS COMPACT(DASDMIGRATE)**
- **SETSYS COMPACT(ALL)**
- **SETSYS ZCOMPRESS(TAPEMIGRATE(YES))**
- **SETSYS ZCOMPRESS(DASDMIGRATE(YES))**
- **SETSYS ZCOMPRESS(ALL)**

For details, see the chapter about DFSMShsm Installation Exit in *z/OS DFSMS Installation Exits,* SC23-6850.

7.3.2 Controlling ZCOMPRESS for Volume Dumps

As part of the availability management DFSMShsm performs two groups of tasks: Dump tasks and backup tasks.

The dump tasks consist of:

- Specifying which volumes to dump, and the dump classes to use
- Specifying when automatic dump processing starts
- Specifying the DFSMSdss DASD I/O buffering technique to use for dump
- Specifying the maximum number of dump tasks
- Specifying the days on which dump occurs
- Specifying the characteristics of dump classes
- Defining dump volumes to DFSMShsm

When zEDC compression services are to be used for DFSMShsm controlled dumps, appropriate settings have to be made in the dump classes. The dump classes to which a volume is dumped determine how the particular dump copies are made and used. You can control the following factors with the dump class:

- Whether the dump tapes are automatically reused when the dumped data is no longer valid
- Whether the dump tapes can be used to restore individual data sets
- The day of the dump cycle on which data will be dumped to the dump class
- What to do with the newly created dump tapes
- How often to dump data to the dump class
- Whether to reset the data-set-changed indicator
- When the data on the dump volumes becomes invalid
- The kind of tape unit to use for the dump tapes
- The number of generations for which copies of the VTOC of dumped volumes will be kept
- The expiration date to use in the tape header label
- The maximum number of dump copies to be stacked on a dump tape assigned to this dump class
- Whether to compress and encrypt the dump data
- Whether the dump class is required or optional

The **DUMPCLASS** parameter of the **DEFINE** command provides the control for the dump class.

A new **DEFINE DUMPCLASS** optional parameter is available:

```
DEFINE DUMPCLASS(ZCOMPRESS(NO | YES))
```

This parameter is valid for **BACKVOL** and **FRBACKUP** when **DUMP** is specified.

> **Remember:** If zEDC hardware is available, the DFSMSdss is started using the **ZCOMPRESS(PREFERRED)** option. In the case of a zEDC hardware failure, the dump might or might not be compressed, depending on the other **DUMPCLASS** options.
>
> The **ZCOMPRESS** and **HWCOMPRESS** keywords are specified through their dump class.

Example 7-2 shows an example of dump class ZEDCTST1 with the **ZCOMPRESS** option set to YES.

Example 7-2 Example DUMPCLASS defined in DFSMShsm

```
DEFINE DUMPCLASS(ZEDCTST1 -
   RETPD(2) AUTOREUSE -
   NORESET -
   UNIT(VT3590) -
   FREQUENCY(0) -
   RETENTIONPERIOD(1)  -
   STACK(2) -
   ZCOMPRESS(YES) -
   DATASETRESTORE -
   VTOCCOPIES(0))
```

Using the Dump Class described previously, we dumped a Storage Group with two volumes assigned. Both volumes were 3390 Mod. 27. At the time of the dump, the volumes had 7% - 9% free space. See Table 7-3 for a listing of the dump results using **ZCOMPRESS(YES)** and **ZCOMPRESS(NO)**.

Table 7-3 Performance of DFSMShsm dumps

	Elapsed Time	Space allocated (on Tape)	# of Tapes used
ZCOMPRESS(YES)	6 Minutes 20 Seconds	36.9 gigabytes (GB)	15
ZCOMPRESS(NO)	8 Minutes 50 Seconds	58.4 GB	28

7.3.3 Recovering data using zEDC

Because DFSMSdss is the data mover, DFSMShsm uses zEDC services to automatically decompress data during a recovery operation, even if the use of zEDC has been disabled using the **SETSYS** or **DEFINE DUMPCLASS** options.

7.3.4 Coexistence of z/OS V1R12 and V1R13

DFSMSdss enables z/OS V1R12 and V1R13 releases to restore backups created using zEDC services. In this case, software inflate is used.

DFSMShsm enables V1R12 and V1R13 releases to **RECALL**, **RECOVER**, **RECOVER from DUMP**, or **FRRECOV from DUMP** data sets migrated, backed up, or dumped using zEDC Services on V2R1. It leverages the coexistence support provided by DFSMSdss.

References

Program temporary fixes (PTFs) for zEDC exploitation or software decompression have a fix category of IBM.Function.zEDC.

See 2.2, “z/OS: Verify the prerequisites” on page 12 for a more detailed reference.

8

zEDC advanced topics

IBM updates IBM 31-bit and 64-bit software development kit (SDK) for z/OS Java Technology Edition, version 7 (5655-W43 and 5655-W44) (IBM SDK7 for z/OS Java). This update is to provide exploitation of the IBM zEnterprise Data Compression (zEDC) Express feature and the Shared Memory Communications over Remote Direct Memory Access (SMC-R), which is used by the 10 gigabit Ethernet (GbE) RDMA over Converged Ethernet (RoCE) Express feature.

IBM Java runtime environment (JRE) V7.0.0 SR7 and version 7 release 1 provide Java applications, IBM Encryption Facility, IBM Sterling Connect:Direct for z/OS V5.2, and IBM WebSphere MQ for z/OS V8 all with the ability to compress data with zEDC Data compression.

IBM z/VM V6R3 delivers support for guest exploitation of the zEDC Express feature on the IBM zEnterprise EC12 (zEC12), IBM zEnterprise BC12 (zBC12), and IBM zEnterprise EC13 (zEC13).

In this chapter, we provide some information about:

- zEDC compression using Java
- IBM Encryption Facility
- WebSphere MQ for z/OS V8
- IBM Sterling Connect:Direct for z/OS V5.2
- z/VM and zEDC
- zlib use by applications

© Copyright IBM Corp. 2015. All rights reserved.

8.1 zEDC compression using Java

The 31-bit SDK for z/OS, Java Technology Edition, V7R1 is the current version of this software development kit (SDK). It is designed to be compliant with the Java Standard Edition 7 (Java SE 7) application programming interfaces (APIs). With 31-bit SDK for z/OS, Java Technology Edition V7R1, you can enable your Java applications to use the zEC12 instruction set.

For more information about z/OS SDK Version 7 Release and more added value IBM content, see the IBM SDK, Java for z/OS, Java Technology Edition, Version 7 Release 1 Information Center on the following website:

`http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/index.jsp`

The following standard hardware and software requirements apply:

- IBM z/OS V2R1 or later operating system.
- zEC12 GA2 or zBC12 server or later.
- zEDC Express coprocessor.
- The zEDC Express software feature must be enabled in an IFAPRDxx parmlib member.

You can use the **`D PCIE`** command to view the current values for the zEDC parameters. Example 2-8 on page 17 shows the output of the command.

To enable compression for Java, perform the following steps:

1. Grant READ access to the FPZ.ACCELERATOR.COMPRESSION resource class to the user ID that will run the Java application. This resource class is a System Authorization Facility (SAF) FACILITY resource class, which regulates access to the zEDC Express coprocessor.
2. Set the z/OS UNIX environment variable, `_HZC_COMPRESSION_METHOD`, to hardware or default. If you set this variable to any other value, zEDC is used.
3. Ensure that the z/OS input buffers for your Java application meet the minimum threshold set by the **`MINREQSIZE`** parameter of the IQPPRMxx PARMLIB member. Otherwise, zlib software compression is used instead. See the IQPPRMxx chapter in *z/OS V2R1.0 MVS Initialization and Tuning Reference,* SA23-1380.
4. Set the Java application to compress files using the java.util.zip.GZIPOutputStream class.

The JRE provides a set of classes under the `java.util.zip` package to perform data compression and decompression. These classes enable users to read, create, and update compressed/uncompressed data using the ZIP, GZIP, or deflate data compression file formats.

Table 8-1 lists some of the frequently used compression API classes in the `java. util.zip` package. The classes listed in this table all use the zlib compression library using the Deflater and Inflater classes to compress and decompress data.

Table 8-1 Frequently used compression API classes in the java.util.zip package

Class Purpose	**Deflater supports general compression using the zlib compression library.**	**Inflater supports general decompression using the zlib compression library.**
InflaterInput- Stream	Reads a stream that is compressed in the *deflate* compression format and decompresses it.	
DeflaterOutput- Stream	Writes compressed data in *deflate* compression format.	
GZIPInput- Stream		Reads a stream that is compressed in the GZIP format and decompresses it.
GZIPOutput- Stream	Writes compressed data in GZIP format.	
ZipInputStream		Reads a stream that is compressed in the ZIP format and decompresses it.
ZipOutput- Stream	Writes compressed data in ZIP format.	

The version of zlib used by IBM Java 7.0.0 SR7 and Java V7R1 on z/OS has been updated to include the changes required to use zEDC with the existing classes.

Figure 8-1 on page 122 shows a sample Java program, which illustrates the use of zEDC. Using GZIPOutputStream, it reads data from one file and then writes compressed data to another file. Note that imports and try/catch logic have been removed for brevity.

In this program, the input buffer for each deflate call is the 16 kilobyte (KB) area represented by buffer, with a 4 KB output area. If the output buffer is not large enough to contain the entire output of a compressed 16 KB input buffer, the gzStream.write blocks processing until all output is processed.

The output buffer size does not affect the decision to use zEDC. It does, however, affect some of the software-based efficiency. Internally, the GZIPOutputStream class keeps calling the Deflate API in a loop to collect all of the output in 4 KB increments. Each call does incur some processing cost, in addition to the extra memory usage to buffer the additional output that exceeds the initial 4 KB output buffer.

The `java.util.zip` classes, in general, provide external means of changing the internal buffer sizes used for both input and output of compression or decompression operations. It can be non-obvious, however, which specific changes need to be made in the Java application to get the benefit of zEDC.

The input buffer size represents the size of the input buffer, which contains the data that needs to be compressed or decompressed. This size determines if zEDC can be used for compression or decompression. If this value is greater than or equal to the threshold, zEDC is used. Otherwise, zlib software-based compression/decompression algorithms are to be used.

The output buffer size represents the size of the output buffer where output data is stored (compressed data for compression or uncompressed data for decompression). The output buffer size provided to the inflate or deflate method does not affect the decision to use zEDC. If this value is smaller than the amount of data in the input buffer, the zEDC code allocates a buffer to hold the overflow output, and subsequent calls to the inflate or deflate methods use the output from this buffer and does not issue additional zEDC requests.

Figure 8-1 shows a sample Java program illustrating the use of zEDC.

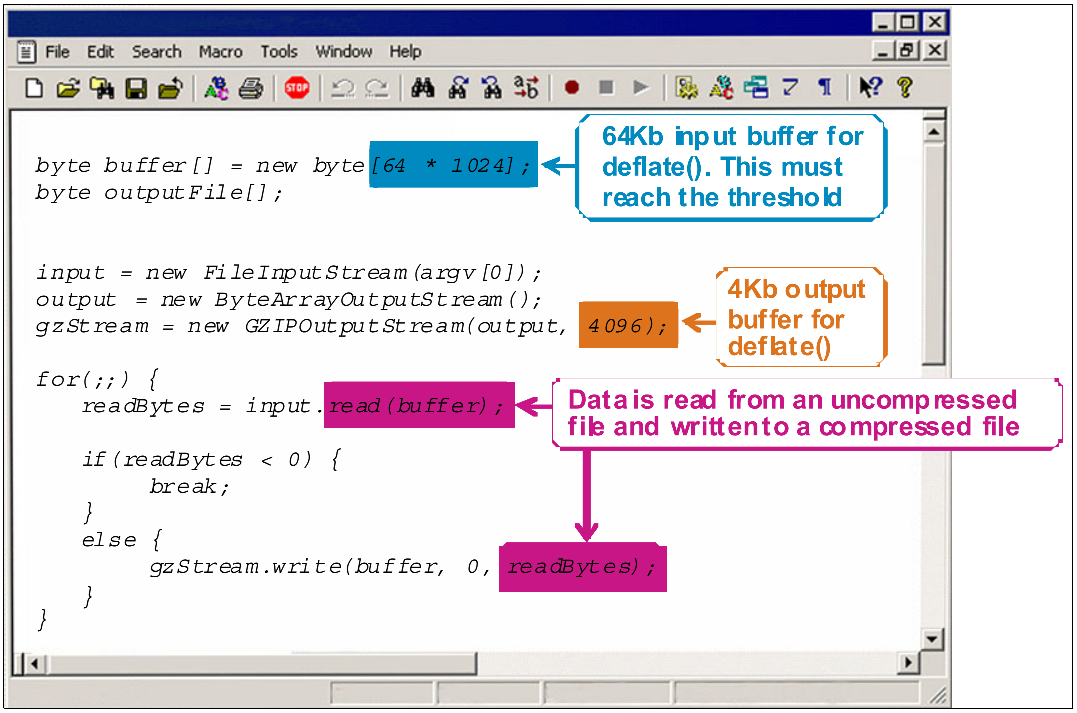


Figure 8-1 Java usage example

Now, look at the Java classes in the `java.util.zip` package and see which buffer values are important. When using the Inflater and Deflater classes directly, the input buffer size is nothing but the size parameter passed using the `setInput` method.

The GZIPInputStream, DeflaterInputStream, and InflaterInputStream classes provide a constructor that enables the input buffer size for the deflate or inflate operation to be specified. The buffer passed to the read method determines the size of the output buffer.The GZIPOutputStream and DeflaterOutputStream classes provide a constructor that enables the output buffer size for deflate and inflate operations to be specified. For these classes, the size of the buffer passed to the write method sets the input buffer size.

The ZipInputStream and ZipOutputStream classes do not provide a constructor that enables the buffer size to be manipulated. For the ZipOutputStream class, the input buffer for the write method can be large enough to qualify for zEDC to be used for compression of the data. The output buffer size is the default, which is 512 bytes. The ZipInputStream always uses 512 byte input buffers, and does not qualify for zEDC.

The JarInputStream class inherits from the ZipInputStream class. The JarOutputStream class inherits from the ZipOutputStream class. They have the same behavior described previously.

For more information about Java on z/OS, see the following website:

http://ibm.com/systems/z/os/zos/tools/java/

8.2 IBM Encryption Facility

IBM z/OS Integrated Cryptographic Service Facility (ICSF) provides an API for z/OS to access cryptographic features and the cryptographic key data sets. IBM Encryption Facility is an application that can use ICSF's APIs and perform all steps necessary to use ICSF:

- Set up a key to be used for encryption.
- Write an application, utility, or script that calls ICSF's services to encrypt your data with a given key.
- Format the output encrypted data into a message that could be understandable to someone for decrypting the information described in the previous bullet.

The Encryption Facility OpenPGP support can be started from either a z/OS UNIX System Services (USS) command prompt, or from a batch job using JZOS. JZOS is a facility within IBM Java that provides the ability to start Java applications from job control language (JCL). JZOS invocation samples are provided with the Encryption Facility product and within the *IBM Encryption Facility for z/OS: User's Guide*, SA23-1349.

The samples consist of three files:

- Procedure in PROCLIB
- Shell script to configure environment variables
- Batch job that calls the sample procedure in PROCLIB

The IBM Encryption Facility, as a Java middleware application, can use the `java.util.zip` classes to compress and decompress data. The IBM Encryption Facility for z/OS V1.2 (function modification identifier (FMID) HCF7740) code has been updated to use proper buffer sizes for `java.util.zip` classes.

This enables the use of zEDC when processing and generating compression OpenPGP (Request for Comments (RFC) 4880) compliant data. IBM Encryption Facility minimum Java requirements to use zEDC are either z/OS, Java Technology Edition, Version 7 Release 1 (Java V7.1) or z/OS, Java Technology Edition, Version 7 SR7.

Example 8-1 shows sample JCL that uses the Java batch program and environment script to encrypt data with IBM Encryption Facility.

Example 8-1 Sample JCL for Java batch program with encryption

```
/*
//JAVA3 EXEC PROC=CSDJZSVM,VERSION='50'
//STDENV DD DSN=<HLQ>.JZOS.JCL(CSDSMPEN),DISP=SHR
//*
//DDDEF DD DSN=HLQ.EFR2.ENC.OUT2,
// DISP=(NEW,CATLG),
// DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760),
// UNIT=SYSALLDA,
// SPACE=(CYL,(5,1))
//*
//MAINARGS DD *
-homedir /etc/encryptionfacility/
-o 'DD:DDDEF'
-rA rsa_md2_4096
-keystore /var/encryptionfacility/keystores/encrdecr/keystore_jceks
-keystore-type JCEKS
-keystore-password password
```

```
-key-password password
-t 'UTF-8'
-e '//HLQ.EFR2.INPUT(CLRTXT)'
/*
```

All IBM Encryption Facility for OpenPGP commands have a syntax where **-homedir** must appear before all of the options, and all of the options must appear before the commands:

```
com.ibm.encryptionfacility.EFOpenPGP [-homedir name] | [options] commands
[arguments]
```

In this command, the following information must be provided:

homedir name Is the name of the `ibmef.config` configuration file that contains specified options to use with the command.

options Is the name of one or more options to use on the command line, and always starts with the Minus sign (-). This option value overrides values in the configuration file.

commands Is the name of one or more commands, and always starts with the Minus sign (-).

arguments Specifies one or more targets of the command, for example, file name, certificate, alias, and so on.

For IBM Encryption Facility environments where compression is already in use, zEDC can provide significant reductions of processor time.

For IBM Encryption Facility environments not already using compression, compression with zEDC can provide reductions of up to 50% in processor time and elapsed time.

Disclaimer: Results are based on internal controlled measurements using IBM Encryption Facility for files containing public domain books. Results might vary by client based on individual workload, data, configuration, and software levels.

zEDC makes it possible to compress the file, using very little processor time before encryption. After the file is compressed, the processor time to encrypt the compressed file is further improved, because there are fewer bytes to encrypt. zEDC hardware compression is expected to use the lowest processor time to produce an encrypted file.

The following list describes the support for hardware compression of OpenPGP messages:

- IBM Encryption Facility for z/OS supports data compression in the OpenPGP message format, when using the passphrase-based encryption (**-c**) command, public key encryption (**-e**) command, and sign (**-s**) command.
- The IBM Encryption Facility decrypt (**-d**) command and verify (**-v**) command support decompression of data in the OpenPGP message format.
- The **-z/COMPRESSION** command option is used to turn on compression when using the **-c**, **-e**, or **-s** commands.
- A compression algorithm name can be specified by using the **-compressname**/**COMPRESS_NAME** command option.

  Supported compression algorithms include ZIP and zlib, which are provided by the IBM Java SDK.
- The **-d** and **-v** commands do not require a command option for compressed data. These commands automatically decompress data in the OpenPGP message format.

zEDC requires a minimum input buffer size for compression and decompression:

- If the input data is smaller than the minimum threshold, the data is processed using traditional software-based compression and decompression.
- Default thresholds are 4 KB for compression and 16 KB for decompression.

  These values can be overridden.
- If the input data is large enough, Encryption Facility uses 324 KB input buffers for compression and 64 KB input buffers for decompression.

8.3 WebSphere MQ for z/OS V8

WebSphere MQ for z/OS V8 uses zEDC for channel message compression. Currently, there are several options for channel message compression, which are specified using the COMPMSG attribute, and two of these, ZLIBHIGH and ZLIBFAST, provide DEFLATE-compliant compression:

ZLIBFAST Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

ZLIBHIGH Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

The COMPMSG(ZLIBFAST) now uses zEDC for compression and decompression, when available.

For more information about WebSphere MQ, see the IBM MQ wesite:

http://www.ibm.com/software/products/en/ibm-mq

8.4 IBM Sterling Connect:Direct for z/OS V5.2

The managed file transfer product IBM Sterling Connect:Direct for z/OS now automatically uses zEDC for file compression and decompression as files are transferred, when the extended compression option is specified. The support is fully compatible with zlib compression used in IBM Sterling Connect:Direct today, so there are no changes required at end points. The only software requirement for Connect:Direct for z/OS is V5R2.

IBM Sterling Connect: Direct for z/OS V5.2 helps client needs with new capabilities to improve performance and security, including the following functions:

- New highly optimized, efficient file compression with zEDC
- New high-speed file transfer interface with the IBM DS8000® line of storage solutions offering improved file transfer rates and reduced Internet Protocol (IP) network usage
- New security and encryption capabilities to help clients meet various internal security and regulatory compliance initiatives, including Federal Information Processing Standard (FIPS), National Institute of Standards and Technology (NIST) SP800-131a, and Transport Layer Security (TLS) 1.2/1.2

Using zEDC for compression over software reduces the elapsed time for file transfers, with a dramatic reduction in processor usage.

For more information about IBM Sterling Connect: Direct for z/OS, see the following website:

http://www.ibm.com/software/products/en/connect-direct

8.5 z/VM and zEDC

z/OS guests running under z/VM V6.3 can use the zEDC Express feature. This can help to reduce disk usage, provide optimized cross-platform exchange of data, and provide higher write rates for System Management Facilities (SMF) data.

z/VM V6.3 support for guest exploitation of RoCE and zEDC adapters is provided with the program temporary fix (PTF) for authorized program analysis report (APAR) VM65417. z/VM support is disabled by default and must not be enabled until driver 15 bundle 21 has been applied. For details about enabling and configuring the z/VM support when the prerequisite bundle is applied, see the following website:

http://www.vm.ibm.com/zvm630/apars.html

The following additional service is also required:

- z/VM 6.3 CMS - APAR VM65437
- z/VM 6.3 TCP/IP - APAR PI20509
- z/VM 6.3 DVF - APAR VM65572
- z/OS 2.1 - APAR OA44482
- z/OS 2.1 - APAR OA43256

8.6 zlib use by applications

The zlib open source library is a lossless data compression library that implements the DEFLATE file format, a variation of the Lempel-Ziv 1977 (LZ77), with software algorithms. The zlib is available for z/OS UNIX System Services (z/OS V2.1) and supports the sending of in-memory compression and decompression request to the zEDC Express.

The zlib data format is itself portable across platforms. The z/OS zlib library is provided as a z/OS UNIX archive file that can be statically linked in applications that currently use zlib, or for exploitation of compression through zEDC Express. Because all of the function signatures are the same, existing zlib-enabled programs can use the zEDC Express.

Applications can use zEDC with industry-standard APIs in the zlib library and Java for z/OS V7.1.

For additional information about zlib, see the following website:

http://zlib.net/

8.6.1 zlib and z/OS

z/OS V2.1 uses the industry-standard zlib open source library available for z/OS UNIX System Services. This version of the library supports the sending of compression and decompression requests to the zEDC Express. The z/OS-provided zlib library is provided as a UNIX archive file that can be statically linked into IBM, independent software vendor (ISV), or client applications that currently use zlib. This enables more use of compression through zEDC Express, and expands potential compression opportunities.

Note that not all of the standard zlib functions are supported using zEDC. For a list of the functions, see Table 27 in *z/OS MVS Programming: Callable Services for High-Level Languages,* SA23-1377. Standard zlib functions and whether they are supported using zEDC are described in the chapter about application interfaces for zEnterprise Data Compression in the same publication.

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 **`buflen`** input parameter 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 **`buflen`** input parameter 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 **`strm`** input parameter is a pointer to a zlib z_stream structure to check. The function returns one of the following values:

  – Integer value of `0` if the stream has gone to the compression accelerator

  – Integer value of `1` if the stream is pending to go to the compression accelerator, but still could fall back to software compression

  – Integer value of `2` if the stream has gone to software compression

  – Value of `Z_STREAM_ERROR` if the stream has not been initialized correctly

8.6.2 Running zlib

To compress data with zEDC, your installation must meet the system requirements. For detailed information, see Chapter 1, “zEDC overview and prerequisites” on page 1.

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 linked into your applications. The following list defines the paths for the zlib archive file and the zlib header files:

   – Path for the zlib archive file: `/usr/lpp/hzc/lib/libzz.a`
   – Path for the zlib header files: `/usr/lpp/hzc/include/`

   > **Requirement:** When a new IBM service is provided for zlib, all applications that *statically* link zlib must relink to use the updated IBM-provided zlib and take advantage of the new function.

2. Provide SAF access:

   Access to zEDC Express is protected by the SAF FACILITY resource class FPZ.ACCELERATOR.COMPRESSION:

   – Give READ access to FPZ.ACCELERATOR.COMPRESSION to the identity of the address space that the zlib task runs in.

3. Use the z/OS UNIX environmental variable, `_HZC_COMPRESSION_METHOD`, to control if zEDC is used for data compression.

   Note: 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. This threshold can be controlled at a system level using the IQPPRMxx PARMLIB member.
5. Allocate the correct amount of storage for input/output (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 the IQPPRMxx PARMLIB member.

When zlib is statically linked into an application that runs on software or hardware that is not compatible with zEDC, zlib uses the compression and decompression, as shown in Figure 8-2.

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.

Figure 8-2 Compression and decompression with zlib

Figure 8-2 also summarizes zEDC error handling:

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

You use IBM MVS callable services for starting unauthorized or System z-authorized interfaces for zEDC. Callable services are for use by any program coded in C, COBOL, Fortran, Pascal, or PL/I, and this information refers to programs written in these languages as high-level language (HLL) programs.

Callable services enable HLL programs to use specific MVS services by issuing program CALLs. For more information, see the description of zEnterprise Data Compression (zEDC) in *z/OS MVS Programming: Callable Services for High-Level Languages,* SA23-1377.

8.7 System z authorized compression services

Although using the API to call zlib is done starting unauthorized interfaces for zEDC, the following compression services are alternatively available when using System z authorized interfaces for zEDC:

FPZ4RZV Rendezvous compression service.

FPZ4PRB Probe device availability compression service.

FPZ4RMR Memory registration compression service.

FPZ4DMR Deregister memory compression service.

FPZ4ABC Submit compression request.

FPZ4URZ Unrendezvous compression request.

System z authorized compression services are described with a usage example in *z/OS MVS Programming: Callable Services for High-Level Languages*, SA23-1377.

A

Additional material

This book refers to additional material that can be downloaded from the Internet, as described in the following sections.

Locating the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Browse to the following website:

`ftp://www.redbooks.ibm.com/redbooks/SG248259`

Alternatively, you can go to the IBM Redbooks website:

`ibm.com/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG24-8259.

Using the web material

The additional web material that accompanies this book includes the following files:

File name	**Description**
`DssDump_.Testrun.Xlsx`	Microsoft Excel spreadsheet of runs results

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space:	100 MB minimum
Operating System:	Windows 7
Processor:	Any
Memory:	4 megabytes (MB)

© Copyright IBM Corp. 2015. All rights reserved.

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material `.zip` file into this folder.

Index

© Copyright IBM Corp. 2015. All rights reserved.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed description of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM zEnterprise EC12 Technical Guide*, SG24-8049
- *Systems Programmer's Guide to: z/OS System Logger*, SG24-6898
- *SMF Logstream Mode: Optimizing the New Paradigm*, SG24-7919
- *Systems Programmer's Guide to: z/OS System Logger, SG24-6898*

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *z/OS MVS Initialization and Tuning Reference,* SA23-1380
- *z/OS MVS Programming: Callable Services for High Level Languages,* SA23-1377
- *z/OS MVS System Commands*, SA38-0666
- *z/OS Hardware Configuration Definition User's Guide,* SC34-2669
- *z/OS MVS System Management Facilities (SMF),* SA38-0667
- *IBM Encryption Facility for z/OS: Planning and Customizing,* SA23-2229
- *IBM Encryption Facility for z/OS: User¡¦s Guide,* SA23-1349
- *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP,* SA23-2230
- *z/OS MVS System Messages, Volume 9 (IGF-IWM)*, SA38-0676
- *z/OS MVS System Messages, Volume 5 (EDG-GFS)*, SA22-7635
- *z/OS MVS System Management Facilities (SMF)*, SA38-0667
- *Resource Measurement Facility Report Analysis*, SC34-2665
- *z/OS MVS Diagnosis: Tools and Service Aids*, GA32-0905
- *DFSMShsm Storage Administration Guide*, SC23-6871
- *DFSMSdfp Storage Administration*, SC23-6860
- *z/OS DFSMS Installation Exits*, SC23-6850

© Copyright IBM Corp. 2015. All rights reserved.

Online resources

These websites are also relevant as further information sources:

- IBM System z Batch Network Analyzer (zBNA) Tool

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132
- Data Extraction Program (CP3KEXTR) for zPCR and zBNA

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4229
- z/OS V2R1 Elements and Features - September 2014

  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/#IEA
- z/OS V2R1 publications can be downloaded from the z/OS Internet Library

  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM Redbooks

Reduce Storage Occupancy and Increase Operations Efficiency with IBM zEnterprise Data Compression

(0.2"spine)
0.17"<->0.473"
90<->249 pages

Reduce Storage Occupancy and Increase Operations Efficiency with IBM zEnterprise Data Compression

Understand zEDC capability and the hardware features

Store compressed data on System z more cost effectively

Leverage zEDC for cross-platform file compression

IBM zEnterprise Data Compression (zEDC) capability and the Peripheral Component Interconnect Express (PCIe or PCI Express) hardware adapter called *zEDC Express* were announced in July 2013 as enhancements to the IBM z/OS V2.1 operating system (OS) and the IBM zEnterprise EC12 (zEC12) and the IBM zEnterprise BC12 (zBC12).

zEDC is optimized for use with large sequential files, and uses an industry-standard compression library. zEDC can help to improve disk usage and optimize cross-platform exchange of data with minimal effect on processor usage.

The first candidate for such compression was the System Management Facility (SMF), and support for basic sequential access method (BSAM) and queued sequential access method (QSAM) followed in first quarter 2014. IBM software development kit (SDK) 7 for z/OS Java, IBM Encryption Facility for z/OS, IBM Sterling Connect:Direct for z/OS and an IBM z/VM guest can also use zEDC Express.

zEDC can also be used for Data Facility Storage Management Subsystem data set services (DFSMSdss) dumps and restores, and for DFSMS hierarchical storage manager (DFSMShsm) when using DFSMSdss for data moves.

This IBM Redbooks publication describes how to set up the zEDC functionality to obtain the benefits of portability, reduced storage space, and reduced processor use for large operational sets of data with the most current IBM System z environment.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, clients, and IBM Business Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8259-00 ISBN 0738440450