OA66536 Publication Updates

DFSMSdfp CDA

03/18/2025

Sushma Sri B

This document describes the updates to the z/OS 3.1 publications as a result of the new function support delivered via OA66536.

1 Overview

The OA66536 APAR delivers new support for existing CDA APIs GDKWRITE and GDKGET. The GDKWRITE and GDKGET APIs are updated for the GDK_BUFFER_DATALOCATION, GDK_PATH_DATALOCATION, GDK_EXIT_DATALOCATION source type to utilize the ability to request zEDC or gzip compression and decompression.

Additionally, the GDKUTIL program is updated to support new keywords COMPRESSION(<compression type>), COMPLEVEL(<compression level>), DECOMPRESS(< compression type>).

1.1 User Actions

In order to start using the new support the user of CDA services, (either through the GDKUTIL program, or an application that invokes the APIs), must take some actions.

1.1.1 Provider File Updates

The CDA sample provider files found in /usr/lpp/dfsms/gdk/samples/providers/ are updated to reflect the enablement of new support. One way to find the updates is to compare your provider file with the equivalent sample provider file. Another way is to take the sample provider file as-is, and update it with the unique values for your cloud provider.

Following is a description of updates and their associated provider file sections.

- The GETOBJECT operation needs to be able to understand metadata tags. In the GETOBJECT operation, a new responseResults array is added. A new entry is added that specifies METAHEADER as the mechanism with a descriptor of "descriptor": "<metadata-header-prefix>" . <metadata-header-prefix> is the appropriate prefix for metadata tags that are returned by the cloud object storage provider. For example, the S3 metadata-header-prefix is x-amz-meta-. For Microsoft Azure object storage, the metadata-header-prefix is x-ms-meta- .
- The GETLARGEOBJECT operation also needs to be able to understand metadata tags when performing the getSize action. In the GETLARGEOBJECT operation, a new JSON object is added to the responseResults array in the “getsize” action, a new responseResults is added in the “data” action. It specifies METAHEADER as the mechanism with a “descriptor” value of "<meta-header-prefix>" . The value of <meta-header-prefix> is described in the above bullet.

1.1.2 New Keyword Exploitation

In order to use the new support, some updates to JCL or an application are required.

1.1.2.1 GDKUTIL invocations

- To compress the data from a UNIX file or data set during Upload, you should write JCL that invokes PGM=GDKUTIL with a SYSIN that specifies UPLOAD COMPRESS(zEDC|gzip). This will use zlib compression services to compress the data using the requested compression algorithm. An optional keyword, COMPLEVEL(MAX|SPEED|DEFAULT|{1-9}), may be specified to further tailor the compression appropriate to your needs.

- To download an object, and have the data decompressed, no additional changes are needed as long as the object has the metadata tag that ends with “zos-compression”, and your provider file has the METAHEADER updates for GETOBJECT and GETLARGEOBJECT. DFSMSdfp CDA will recognize the metadata tag and decompress the data as it is received.
    - If you know that the object has data compressed by the zEDC or gzip algorithm, but either your provider file doesn’t have the METAHEADER description in the GETOBJECT/GETLARGEOBJECT operations, or the object does not have the metadata tag, you can specify the DECOMPRESS keyword to cause DFSMSdfp CDA to perform the decompression. If you are not sure of the type of compression algorithm used, you may specify DECOMPRESS(zlib) to request that the imbedded zlib headers be used to figure out the algorithm to use. If the zlib headers do not exist in the object, you can specify the algorithm type to use with DECOMPRESS(zEDC) or DECOMPRESS(gzip).
    - If you know that the object has compressed data, but do not want it to be decompressed, you may specify DECOMPRESS(NONE) to download the object as-is without any further processing.

1.1.2.2 Application usage

In order to use the new functionality from an application, new optional parameters are required.

- The GDKWRITE API is updated to recognize new optional parameters.
    - “compression” can be passed with a value of “zEDC” or “gzip” to request that the data be compressed before sent to the cloud object storage.
    - “compLevel” can be passed with a value of “SPEED”, “MAX”, “DEFAULT”, or a number from 1 to 9. i.e. “3”
    - “Get-Sent-Data-LengthE" can be passed with the value being a pointer to an 8-byte number field where total number of bytes sent to the server will be saved.
- The GDKGET API is updated to recognize two new optional parameters.
    - “decompress” can be passed with the value of “true”, “false”, or “attempt-inflate”.
    - “compression” can be passed with the values of “zEDC”, “gzip”, “zlib”, or “NONE”

2 Publication Updates

2.1 z/OS MVS Programming: Callable Services for High Level Languages

MVS Programming: Callable Services for High Level Languages

SA23-1377

Part 11. Cloud Data Access Services is updated as follows:

2.1.1 Chapter 26. Cloud Data Access(CDA) API basics

Chapter 26. Cloud Data Access (CDA) API basics – Elements of Cloud Data Access is updated to add the following new bullets:

Compress an object before writing to cloud storage.

Decompress an object before downloading it to z/OS.

Chapter 30. Cloud Data Access callable services is updated as follows:

2.1.1.1 GDKGET — Retrieve a cloud object

The Optional Parameters table is updated to add the following rows:

compression	Character	String with one of the following options: • zEDC – DFSMSdfp CDA will use the zEDC compression algorithm to decompress the data from the object. • gzip – DFSMSdfp CDA will use the gzip compression algorithm to decompress the data from the object. If this parameter is not passed, DFSMSdfp CDA will look for the "zos-compression" metadata tag to know whether to decompress data and which algorithm to use. If the object does not have the "zos-compression" metadata tag and the "compression" optional parameter wasn't passed, DFSMSdfp CDA will not attempt to decompress the data. (Note the "decompress" optional parm may override this.) The GETOBJECT and GETLARGEOBJECT operations must have a METAHEADER entry in the responseResults array that describes the prefix for a metadata header.
decompress	Character	String with one of the following options: • "true" indicates that DFSMSdfp CDA will attempt to decompress the data from the object. If the data is not compressed, the operation is stopped with a return code 149 (GDK_DECOMPRESSION_FAILURE). • "false" indicates that DFSMSdfp CDA will not attempt any decompression, regardless of the existence of the "zos-compression" metadata tag on the object. • "attempt-inflate" indicates that DFSMSdfp CDA should attempt decompression of the data. If an error occurs due to the data not being compressed, processing will continue with the object data being given to the caller as-is. If this optional parameter is not passed, DFSMSdfp CDA will examine the object's HTTP headers for a "zos- compression" metadata tag. If the object doesn't have the zos-compression metadata header or the responseResults array doesn't contain a METAHEADER description, then no decompression will be performed

Return and reason codes

Return code	Constant Name	Explanation
145	GDK_ICONV_ERROR	Conversion error, converting from one code page to another code page.
148	GDK_DECOMPRESSION_INIT_FAILURE	An error occurred while initialising the stream to perform decompression.
149	GDK_DECOMPRESSION_FAILURE	An error occurred while decompressing the data.
151	GDK_NO_MEMORY_AVAILABLE	Could not allocate memory for decompressing the data.
152	GDK_INVALID_COMP_PARMS	Invalid decompress or compression parameter passed.

2.1.1.2 GDKWRITE – Send a cloud object

In the description of the optionalParmStructPtr, the Optional Parameters table is updated to add the following rows:

compression	character	String with one of the following options: • zEDC – DFSMSdfp CDA will use the zEDC compression algorithm to compress the data sent to the object. • gzip – DFSMSdfp CDA will use the gzip compression algorithm to decompress the data sent to the object. If the WRITEOBJECT and WRITELARGEOBJECT operations in the provider file have a METAHEADER description in the requestParameters, a metadata tag with the suffix “zos-compression" and a value indicating the type of compression used (zEDC or gzip) will be associated with the object.
compLevel	character	String with one of the following options: SPEED - perform fastest compression with minimal CPU usage; lower compression ratio. MAX – Maximizes compression ratio with more CPU DEFAULT – Performs efficient compression, maintaining speed. {1-9} – A number indicating the compression level to be used when initializing compression through the zlib APIs.
Get-Sent-Data-LengthE	Address	The address of an 8-byte number field where total number of bytes sent to the server will be saved after successful completion.

Return and Reason Codes

Return code	Constant Name	Explanation
145	GDK_ICONV_ERROR	Conversion error, converting from one code page to another code page.
148	GDK_COMPRESSION_INIT_FAILURE	An error occurred while initialising the stream to perform compression.
149	GDK_COMPRESSION_FAILURE	An error occurred while compressing the data.
151	GDK_NO_MEMORY_AVAILABLE	Could not allocate memory for compressing the data.
152	GDK_INVALID_COMP_PARMS	Invalid compression or compLevel parameter passed.

2.2 z/OS DFSMSdfp Utilities

SC23-6864-60

Chapter 2: GDKUTIL (Cloud Object Utility) Program, is updated. In the SYSIN Statement table, make the following changes:

In the Keywords section, add the following rows in the appropriate place:

COMPRESSION(zEDC|gzip)	When specified on the UPLOAD command, this indicates that DFSMSdfp CDA should perform the requested compression algorithm on the data sent to the cloud object. If a METAHEADER description exists in the requestParameters array for the WRITEOBJECT and WRITELARGEOBJECT operations, then a metadata tag for "zos-compression" with the value of the requested algorithm. • zEDC – Compress using the z Enterprise Data Compression algorithm. • gzip – Compress using the gzip compression algorithm. May be shortened to COMPRESS.
COMPLEVEL(MAX|SPEED|DEFAULT|{1-9})	• MAX indicates that CDA should try to get the most compression for data being sent. It can result in extra CPU usage as the best compression result is attempted. • SPEED indicates that CDA should request the compression performed be done as quickly as possible, even if the compression ration is not that good. • DEFAULT indicates that the default compression level should be used. It is a mid-point between SPEED and MAX. • 1-9 – A number that indicates the specific compression level to be given to the zlib interface.
DECOMPRESS(zEDC|gzip| zlib|NONE)	When specified on the DOWNLOAD command, this indicates that DFSMSdfp CDA should decompress the cloud object data using the requested algorithm, regardless of the existence of the "zos-compression" metadata tag. • zEDC – Decompress using the zEDC algorithm • gzip – Decompress using the gzip algorithm • zlib – Use the zlib headers imbedded in the object data to determine the algorithm to use. • NONE – Do not perform any decompression, regardless of the existence of the zos-compression metadata tag on the object. If not specified, but the object has the "zos-compression" metadata tag, DFSMSdfp CDA will decompress the data according to the value of the metadata tag.

New examples are added to the **GDKUTIL Examples** section as follows:

Example 12: Compress an object

In this example, the GDKUTIL utility is used to compress the object with the zEDC algorithm and upload it to a cloud.

```
//CRBUCKET EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
  UPLOAD COMPRESSION(ZEDC)
  PROVIDER(IBMCOS)
  LOG(DEBUG)
/*
//OBJNAME  DD  *
   /bucket-name/multi01/dir1/CavesofTerrorpg18970.txt
/*
//LOCNAME  DD  *
 /OA66536/books/CavesofTerrorpg18970.txt
/*
```

Example 13: Decompress and Download the object

In this example, the GDKUTIL utility is used to download an object. The user knows that the object has data compressed with the zEDC algorithm, and the object does not need have the zos-compression metadata tag attached to it.

```
//DOWNCOMP EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
 DOWNLOAD PROVIDER(IBMCOS) DECOMPRESS(zEDC)
/*
//OBJNAME  DD  *
   /bucket-name/multi01/dir1/CavesofTerrorpg18970.txt
/*
//LOCNAME  DD  *
 /OA66536/downloads/CavesofTerrorpg18970.txt
/*
```