OA63892 Publication Updates

S3 Support for TCT

DFSMSdss

04/26/2024

Ernesto Figueroa

This document describes the updates to the z/OS 3.1 publications as a result of the new function delivered via OA63892.

1 Overview

The OA63892 APAR support enhances the existing DFSMSdss transparent cloud tiering (TCT) offering to assist in resolving the single point of failure (HMC) via the use of Cloud Data Access (CDA).

When the cloud is TAPE-OBJECT (TS7700), a CDA provider file may specify multiple HMC endpoints. The endpoints will be obtained at random allowing workloads to be spread across the specified endpoints. If an endpoint is not reachable during DFSMSdss initialization, we will attempt the other endpoints in the list until the list is exhausted, or a successful connection is made.

For all other cloud types, A CDA provider file will describe a cloud connection and tailorable supported operations for CDA to perform directly with the object storage cloud instead of via an HMC proxy. DFSMSdss will request meta-data movement to be performed via CDA while user-data is still moved via TCT.

1.1 User Actions

In order to invoke this support, the user must take several actions.

1.1.1 New Function Exploitation

In order to use the new functionality, a new keyword must be specified. If the keyword is not specified, there will be no behavior changes as DFSMSdss will instead default to the existing TCT functionality.

1.1.1.1 New Keyword: CDAPROVIDERFILE

A new keyword, CDAPROVIDERFILE, or its supported abbreviation, CDAPROV, must be specified in the JCL.

In order to verify that the new support was used, users can look in the listing for the following message:

```
ADR618I (R/I)-RI03 (01), CLOUD INFORMATION OBTAINED FROM CDA
```

1.1.1.1.1 ADRPATCH offset ‘62’x

An installation may decide they want the new enhancement to be the default option in lieu of specifying the CDAPROVIDERFILE keyword. New patch byte at offset ‘62’x may be set to a value greater than 0. DFSMSdss will use the new support regardless of whether the keyword is specified.

1.1.2 The Provider File

This enhancement requires the user invoking DFSMSdss to have an OMVS segment because a CDA provider file is necessary for this new functionality. The provider file is a JSON format file, that contains information about how to connect to the cloud. It must be named in the form '*cloud_name*. json' such that the DFSMSdss command is specifies CLOUD(*cloud_name*).

Sample provider files for both cloud types are shipped by DFSMSdss and can be found in '/usr/lpp/dfsms/dss/samples/'. CDA also ships sample files, however there are additional required fields that are necessary for them to be compatible with DFSMSdss.

2 Storage Administration

2.1 z/OS DFSMS Storage Administration Guide

2.1.1 Requirements for running DFSMSdss

2.1.1.1 Storage requirements

A new paragraph should be added after the PARALLEL command description and before Table 1:

> When performing a DUMP FULL to cloud by using the Cloud Data Access (CDA) support, an additional 8 megabytes of overhead is required due to CDA.

The section that describes 'dsssize' should be updated from 2065KB to 2550KB:

> dsssize
>
> DFSMSdss load module size, 2550KB.

2.1.2 DFSMSdss patch area

A new link for "Check for a CDA Provider file regardless of the CDAPROVIDERFILE keyword's presence"

The following description should be included for the new patch area:

The CDAPROVIDERFILE keyword must be specified in order for DFSMSdss to check for a CDA Provider file. A patch byte can be specified so that DFSMSdss will default to always check for a CDA provider file regardless of whether the CDAPROVIDERFILE keyword was specified or not.

The function is affected by setting the flag at offset X'62' in ADRPATCH. The settings are as follows:

X'00'

DFSMSdss functions normally. DFSMSdss will only look for a CDA provider file when the CDAPROVIDERFILE keyword is specified.

Any setting other than X'00'

DFSMSdss will always check for a CDA provider file regardless of the CDAPROVIDERFILE keyword's presence.

To set the flag to on dynamically, use the SET PATCH command.

2.1.2.1 ADRPTCHB data area (UPDATE)

The table should be modified to the following:

87	(57)	UNSIGNED	1	PBMSGTIME	ADD TIMESTAMPS TO MESSAGES IN EACH OF THE VARIOUS CLASSES
88	(58)	UNSIGNED	3	*	RESERVED
91	(5B)	UNSIGNED	1	PBEATRNV	Set EATTR=OPT for nonVSAM allocations
92	(5C)	UNSIGNED	1	PBZCFAIL	SET for ZCOMPRESS RESTORE ERROR
93	(5D)	UNSIGNED	1	PBCLWDCTNR	When creating a backup to an object storage cloud Allow existing objects to be overwritten.
94	(5E)	UNSIGNED	1	PBBYFCWDEN	FCWD ENHANCEMENT BYPASS
95	(5F)	UNSIGNED	1	PBCLONERESET	ALLOW RESET WITH CLONE
96	(60)	UNSIGNED	1	PBVTOCERR	Info message on VTOC error
97	(61)	UNSIGNED	1	PBSGWARN	Omit warning message for unavailable SG volumes
98	(62)	UNSIGNED	1	PBCLWDPROV	Check for CDA provider file regardless of CDAPROVIDERFILE keyword's presence

2.1.3 Managing Availability with DFSMSdss

2.1.3.1 Backing up data sets

2.1.3.1.1 Backing up data sets to an object storage cloud (UPDATE)

This section should be changed to reflect the following:

DFSMSdss can create logical data set backups to an object storage cloud by using an IBM DS8000 Storage solution called transparent cloud tiering (TCT). Using this functionality results in server-less movement between the z/OS Host and the DS8000. TCT translates into significant savings in CPU utilization within z/OS. Since this process eliminates data movement from the host, DFSMSdss is unable to perform any data manipulation. The following types of processing require data manipulation:

- Reblocking: Reblocking occurs when you specify the REBLOCK keyword or when the VTOC indicates that the data set can be reblocked. The REBLOCK keyword and the indicator in the VTOC is ignored when a backup is being restored from an object storage cloud.

- PDS compression: DFSMSdss compresses a PDS data set during backup processing, by default. You can specify the NOPACKING keyword to prevent DFSMSdss from compressing the PDS. There is no special action needed in order to backup a PDS to an object storage cloud. When creating a backup to an object storage cloud, DFSMSdss operates as if NOPACKING(**) is specified.
- Changing stripe counts: The source stripe count must be the same as the target stripe count for a striped extended format data set. When restoring a backup from an object storage cloud, DFSMSdss ensures that the stripe count does not change. If a data set cannot be created with the same number of stripes the allocation will fail and the data set will not be restored.
- An individual stripe extending to more than one volume: DFSMSdss cannot backup striped VSAM data sets when one or more of the stripes spans volumes. DFSMSdss also cannot backup a single striped version 1 extended format sequential data sets that are multivolume.
- Block-by-block processing of direct access data sets: Block-by-block processing occurs when you specify the RELBLOCKADDRESS OR the AUTORELOCKADDRESS keyword. When restoring a backup from an object storage cloud, DFSMSdss operates as if RELBLOCKADDRESS and AUTORELBLOCKADDRESS are not specified.
- Compression and Encryption – DFSMSdss is unable to perform compression or encryption of its buffers since the data movement is being offloaded to the DS8000 storage. This includes the ZCOMPRESS keyword.

The following sections will be moved to new section “Using cloud object storage”

- Designating cloud as a backup target
- Managing backups created in the cloud
- Invoking from an application program

2.1.3.1.2 Using Cloud Object Storage (UPDATE)

The following subsections from the current version of “Backing up data sets to an object storage cloud” will be moved to this section:

- Designating cloud as a backup target
- Managing backups created in the cloud (no changes to this subsection)
- Invoking from an application program

In addition, the following subsections will have the following changes (subsections bolded):

Designating a cloud as a backup target (UPDATE)

The text of this subsection should be changed to the following:

DFSMSdss can write to cloud object storage when the following requirements exist:

- The source device is in a storage controller that supports transparent cloud tiering.
- The source device is in a storage controller that is connected to an object storage cloud.
- The storage controller and z/OS host are both connected to the same object storage cloud.

The CLOUD(cloud_name) keyword tells DFSMSdss that the backup is to be directed to an object storage cloud. The information describing the target object storage cloud is determined in the following order:

1. z/OS Cloud Data Access (CDA) Provider File will be used if CDAPROVIDERFILE keyword is specified and a valid CDA provider file with the name ‘cloud_name.json’ is found.
2. Otherwise, the SMS Network Connection will be used if the cloud_name exists in the active SMS configuration.

DFSMSdss uses the CDA provider file or the SMS network construct to obtain the information necessary to make a connection to the object storage cloud. When the CLOUD keyword is specified, the following keywords must also be specified:

- CONTAINER(container_name): this keyword specifies the container in which the objects are to be stored. The container specified as input does not have to exist at the time of the backup. If necessary, DFSMSdss will create it for you. The container name specified will be prefixed with the string ‘SYSZADR.’ when invoked via batch JCL. This is to distinguish DFSMSdss backups.

- OBJECTPREFIX(object_prefix): this keyword specifies a unique prefix that DFSMSdss is to use for all the objects that make up a particular backup. The prefix provides uniqueness among multiple backups in the same container. One way to ensure uniqueness among multiple backups of the same data is to use timestamps as part of the object prefix. At the beginning of the backup process, DFSMSdss will determine if a backup using the same object_prefix already exists in the specified cloud_name and container_name. If a backup exists with the same object_prefix , then DFSMSdss will fail the backup. To overwrite a backup using the same object_prefix, you can set a patch at offset x'5D'.

When using an SMS network connection information, one of the following keywords is required:

- CLOUDCREDENTIALS(cloud_credentials): this keyword specifies the password for the account that is defined within the cloud construct. This password is case-sensitive and is suppressed from SYSPRINT and SVCDUMPs. The cloud credentials must be kept secure. If there are batch JCL jobs that specify this keyword, then those jobs should be in a data set or library that has limited access and controlled by a security product. If the credentials cannot be kept secure, then do not use this keyword. Using the CDACREDSTORE keyword instead of the CLOUDCREDENTIALS keyword is recommended.

- CDACREDSTORE: this keyword specifies that the credentials for the specified cloud_name were encrypted and stored by the z/OS Cloud Data Access Authorization Utility.

CDA Provider File (NEW)

The information in section 3.3.2 “DFSMSdss Sample Files” should be included in this section of the publication

This subsection should include the following text:

For more information about the setup and configuration of z/OS Cloud Data Access (CDA) to include provider file, key file updates, and all CDA topics, see Cloud Data Access (CDA) Services in z/OS MVS Programming: Callable Services for High-Level Languages.

The CDAPROVIDERFILE keyword must be specified for DFSMSdss to check for a CDA provider file. When the CDAPROVIDERFILE keyword is specified, DFSMSdss first checks for a CDA provider file that matches the specified cloud name. If one does not exist or the CDA provider file is empty, then the information in the SMS Network connection is used to communicate with the object storage cloud.

When DFSMSdss is obtaining information from a CDA provider file, the following fields are required:

enableDFSMSdss

This key identifies whether or not DFSMSdss can utilize this provider file.

A key value other than YES or the absence of this key will result in the provider file being ignored.

tctType

When the value is set to TAPE-OBJECT the cloud being described is a TS7700 Advanced Object store. All data and utility operations flow through the DS8000 storage cloud proxy using either FICON or Ethernet connections.

When the value is set to TCT all user-data flows through the DS8000 storage subsystem cloud proxy using FICON. All meta-data and utility operations flow directly with the storage cloud via Ethernet instead of using the DS8000 proxy.

tctIdentity

This key must have the value of the DS8000 HMC service account ID; the same value that would have been specified in the Identity field of the SMS network connection.

supportedOperations

This key must be present as an empty list (i.e []) or a list of operations.
When the tctType is not TAPE-OBJECT, the following operations must be present:

- GETOBJECT
- GETLARGEOBJECT
- WRITEOBJECT
- WRITELARGEOBJECT
- LISTOBJECT
- DELETEOBJECT
- CREATEBUCKET
- DELETEBUCKET
- LISTBUCKETS

port

When tctType is TAPE-OBJECT, the port must be specified either in this key or as part of the host list. When tctType is not TAPE-OBJECT, it is not required to specify port since CDA may default to a port they identify.

sslVersion

When tctType is TAPE-OBJECT, if sslVersion is specified, sslKey must also be included. Otherwise, they can both be omitted

Note: If the port key and sslKey are specified in the hostname, they take precedence over their key/value pair counterparts.

When tctType is TAPE-OBJECT, If sslVersion is specified, sslKey must also be included. Otherwise, they can both be omitted. If the port key and sslKey are specified in the hostname, they take precedence over their key/value pair counterparts.

The following key is optional:

cloudName

If cloudname is not specified, the cloud name is assumed to be the name of the provider file. If you would like to use a provider file name that is different from your provider file you must specify the key/value cloudName with a cloud name known to the DS8000.

A sample CDA provider file with the required DFSMSdss JSON key/value pairs can be found in /usr/lpp/dfsms/dss/samples. This is provided as a simple example; the files in /usr/lpp/dfsms/cda/ should be used as prevailing examples for future enhancements.

S3CLOUD.json

This provider file assumes that the DS8K has a cloud name S3CLOUD. This is a 'true' cloud. For any cloud type that CDA supports, use this example.

TAPECLWD.json

This provider file assumes that the DS8K has a TAPE-OBJECT cloud named TAPECLWD. Use this provider file example for HMCs available for serving as a proxy to communicate with the TS7700.

Invocation from an application program (existing)

The following paragraph should be added at the end of the section:

When utilizing CDA, DFSMSdss will ask CDA to ignore the invokers cda configuration file (located in ~/gdk/config.json. CDA will not issue debug content to the SYSOUT DD unless DEBUG(CLDMSG(DETAILED)) is specified.

2.1.3.2 Backing up volumes (existing)

2.1.3.2.1 Physical Volume Dump (existing)

A new subsection called “Physical volume dump to an object storage cloud” should be added, with the following text:

When creating a backup in object storage cloud, the CLOUD specification is used instead of the OUTDDNAME keyword. The OUTDDNAME and the CLOUD keywords are mutually exclusive.

It is currently not possible to perform a physical data set restore from a full volume image when it is in object storage cloud.

2.2 z/OS DFSMSdss Storage Administration Reference

2.2.1 Syntax – DFSMSdss Function Commands

2.2.1.1 Additional keywords update for DUMP, RESTORE, and CLOUDUTILS when CLOUD is specified

The command syntax diagram should be updated to reflect a new additional keyword CDAPROVIDERFILE(CDAPROV) when CLOUD is specified.

2.2.1.2 CLOUD keyword for DUMP, RESTORE, and CLOUDUTILS commands (UPDATE)

The following note, should be added to the existing note box:

- DFSMSdss will look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name if CDAPROVIDERFILE keyword is specified.

2.2.1.3 CDAPROVIDERFILE keyword for DUMP, RESTORE, and CLOUDUTILS commands (NEW)

Specifies that DFSMSdss should look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name. If a CDA provider file is not found, DFSMSdss will default to an SMS Network Connection.

Note:
• CLOUD keyword is required • If a valid CDA provider file is empty, then the information in a SMS Network connection will be used • For further details see CDA Provider File section (link) • This keyword may be set to a default specification using ADRPATCH offset x'62'.

2.2.1.4 CLOUD Keyword for DUMP, RESTORE, and CLOUDUTILS Commands (UPDATE)

For each of the command keyword descriptions for CLOUD, the explanation should be updated to reflect the following text:

ccn

Specifies the name of either a CDA provider file or an SMS construct that identifies the cloud storage the dump is to be written to. If a CDA provider file by the name of ccn.json does not exist or is empty, a network connection construct by the name of ccn must exist in the active SMS configuration.

DFSMSdss uses either the provider file or the SMS construct to obtain the information that is needed to make a connection to the object storage cloud.

Note 2 should be updated to reflect the following, and broken up into two bullets:

- When a network connection construct is found, you must specify either the CLOUDCREDENTIALS or CDACREDSTORE keywords.
- You must specify the CONTAINER and OBJECTPREFIX keywords when specifying the CLOUD keyword.

2.2.1.5 CONTAINER Keyword for DUMP, RESTORE, and CLOUDUTILS Commands (UPDATE)

The note box should be changed to reflect the following:

1. The name can be up to 128 characters in length. If a CDA provider file was identified, the container name may be folded to lowercase.

2. The allowable characters are uppercase letters A-Z, as well as numbers 0-9 and special characters $ @ # - _ and . in the non-first position.

3. Do not specify CONTAINER during DUMP with OUTDD and CONCURRENT, COMPRESS, HWCOMPRESS, ENCRYPT, ZCOMPRESS, RSA or FCWITHDRAW. If you specify VALIDATE or OPTIMIZE those keywords will be ignored.

4. When a network connection is found, you must specify the CLOUDCREDENTIALS or CDACREDSTORE keyword.

5. You must specify the CLOUD and OBJECTPREFIX keywords when specifying the CONTAINER keyword.

2.2.1.6 LIST Keyword for CLOUDUTILS Command (UPDATE)

The first bullet should be updated to reflect the following:

- If CONTAINER is not specified, the request is to list all DFSMSdss-created containers. If a CDA provider file was identified, the container names may be printed in lowercase.

2.2.1.7 CLOUDUTILS command for DFSMSdss

Add the following paragraph at the end:

- When utilizing Cloud Data Access (CDA), the container name may be folded into lowercase.

3 z/OS MVS System Messages, Vol 1 (ABA-AOM)

The ‘ADR Messages’ section of this publication will be updated for the following message(s):

3.1 ADR610E (existing)

Add the following text:

Explanation

5 A non-empty CDA provider file matching the specified cloud was found, but the provider file is missing a necessary field, or a field has invalid information. The missing or incorrect field is identified.

Programmer response

For reason code 5, if the intent was to use the CDA provider file, update the provider file to provide the correct or missing information. Otherwise, delete or rename the CDA provider file.

3.2 ADR617E (new)

ADR617E(ttt)-mmmmm(yy), Z/OS CLOUD DATA ACCESS ENCOUNTERED AN ERROR WHILE PERFORMING A *cda_service* SERVICE, RETURN CODE *cda_rc*

[RETURN CODE TRANSLATION: *explanation*]

[FOR OBJECT: *object_name*]

[OPERATION NAME: *op_name*]

[HTTP RESPONSE CODE: *http_code*]

[HWTHRQST RETURN CODE: *hwth_code*]

[HWTHRQST DIAGAREA: *diagarea]*

[HTTP RESPONSE BODY: response_body]

Explanation

A failure occurred while performing the identified Cloud Data Access service (*cda_service*). The *cda_service* completed with the specified return code (*cda_rc*). The message may be followed by one or more of the following pieces of additional information.

- *explanation* – The CDA service return code (cda_rc) text translation.
- *object_name* – The object name that was being processed at the time of error
- *op_name* – The identified service was GDKGEN, this field identifies the operation that was requested
- *http_code* – The service performed the operation and resulted in this HTTP response code error. Value is presented in hexadecimal.
- *hwth_code* – CDA invoked the z/OS Client Web Enablement Toolkit HWTHRQST service. The service returned this error. Value is presented in hexadecimal.
- *diagarea* – This is the z/OS Client Web Enablement Toolkit HWTHRQST service DiagArea.
- *response_body* – The first 100 bytes of the HTTP response body.

System Action

This error precludes the use of cloud storage. The return code is set to 8.

Operator Response

None.

Programmer response

Refer to the identified z/OS Cloud Data Access callable service and associated return code in z/OS MVS Programming: Callable Services for High Level Languages to identify the reason for the error.

If HWTHRQST areas are provided, refer to the z/OS Client Web Enablement Toolkit section in z/OS MVS Programming: Callable Services for High Level Languages to identify the reason for the error.

3.3 ADR618I (new)

ADR618I(ttt)-mmmmm(yy), CLOUD INFORMATION OBTAINED FROM {SMS | CDA}

Explanation

When CDA: a valid CDA Provider File was found and used to obtain cloud information

When SMS: DEBUG(CLMSG(SUMMARY)) or higher was specified and cloud information was obtained via SMS

System action

Processing continues.

Operator response

None.

Programmer response

None.

Source

DFSMSdss