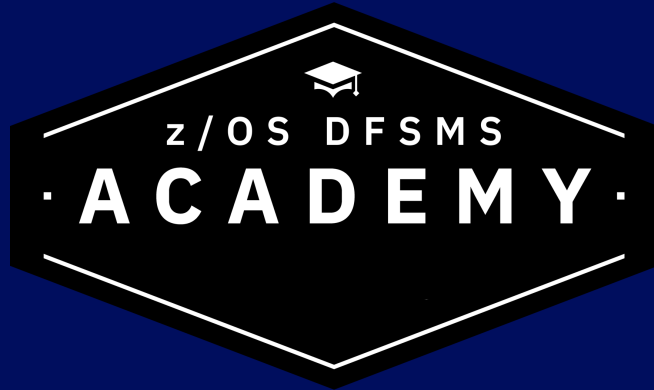


# DFSMSdss Deep(er) Dive Data Movement

Carly Fife  
DFSMSdss Development  
[carly.fife@ibm.com](mailto:carly.fife@ibm.com)

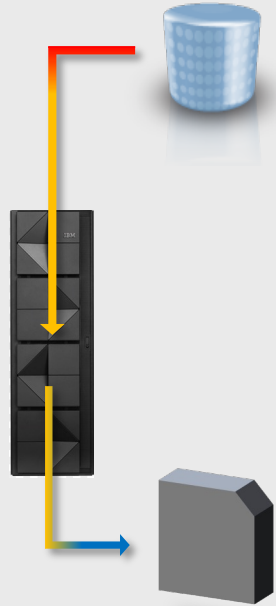
Robert Gensler  
STSM-DFSMSHsm Product Owner  
[rgensle@us.ibm.com](mailto:rgensle@us.ibm.com)



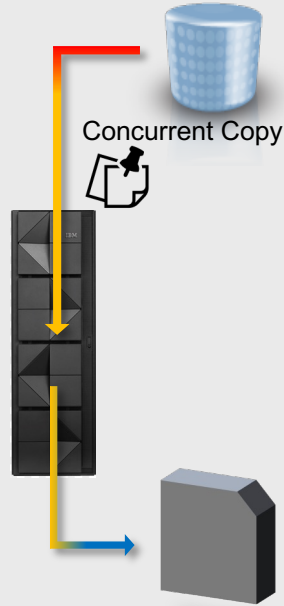
# What are we going to chat about over the next 90 minutes?

- Data movement within the online storage tier
  - Basic APIs
  - Advanced APIs
- Data movement between the online and offline storage tiers
  - Transparent Cloud Tiering
- Converting you existing JCL to JCL using TCT

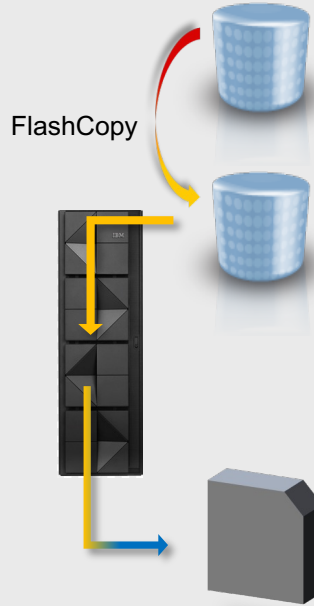
# Evolution of Data Movement



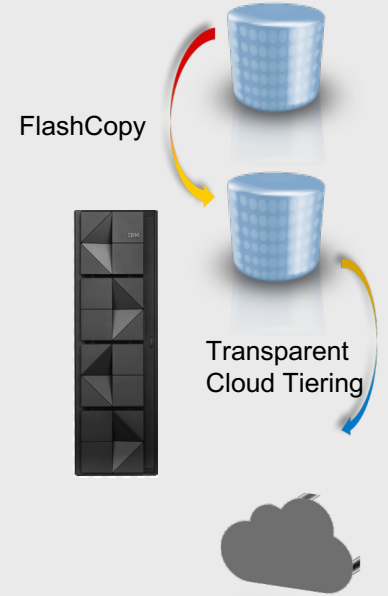
Classic Volume Dump



Concurrent Copy  
Volume Dump

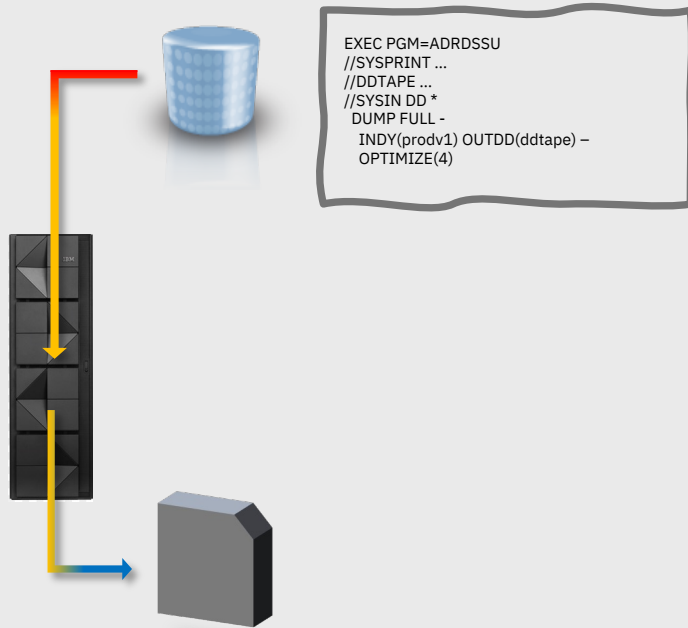


Volume Dump of  
FlashCopy target



Volume Dump of  
FlashCopy target  
using TCT

# Classical Volume Dump



By default, DFSMSdss will read a single tracks worth of data at a time.

Specifying OPTIMIZE(4) will cause DFSMSdss to read up to a cylinders worth of data with a single read.

Uses more memory, but improves throughput and performance of the DUMP operation

# Concurrent Copy Overview



$t_0$
$t_1$
$t_0$
$t_0$
$t_0$
$t_0$
$t_0$

1	
0	$t_0$
1	
1	
1	
1	
1	

Concurrent Copy is a point in time copy technique that allows DFSMSdss to backup data while the application is still reading and writing to it. DFSMSdss requests a new session is to be established for a set of tracks from a volume at a point in time called  $t_0$

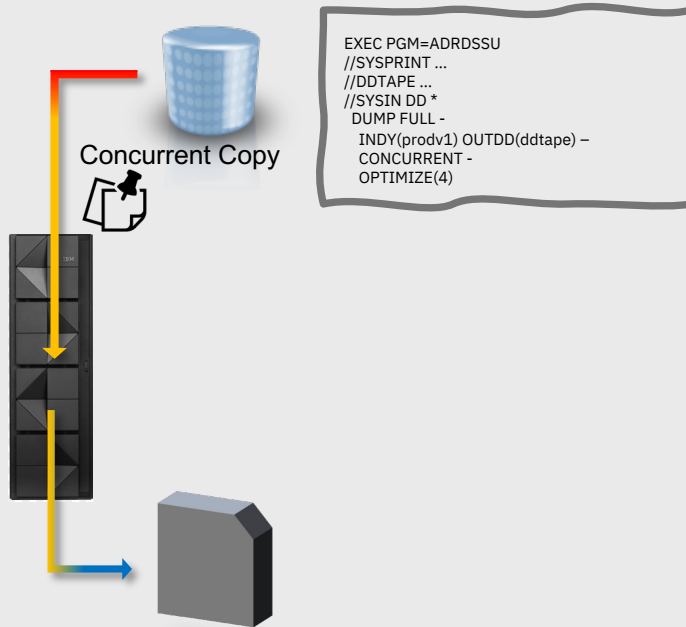
When data that is part of a CC session is updated, the storage system must keep the original 'time<sub>0</sub>' version of the data for DFSMSdss to be able to read later. This version of data is kept in cache either in the storage system or in the z/OS host

A Concurrent Copy operation is a two-phase process

- Logical establish
- Physical data movement

Once the CC session is established applications can resume access to the data sets on volume A while DFSMSdss is creating the DUMP by reading tracks from that concurrent copy session.

# Concurrent Copy

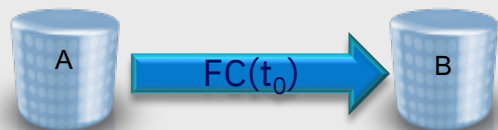


There are two cases to consider:

- DFSMSDss requests a read of data that has not been updated by the application
  - The read request is serviced directly from the tracks on the volume
- DFSMSDss requests a read of data that has been updated by the application
  - The read request cannot get the data directly from the tracks on disk because it has been updated to a t<sub>1</sub> version
  - The read request must receive the t<sub>0</sub> version from cache

If there is a lot of update activity to the data while the backup is in progress, this can put a lot of stress on the cache of the storage system as well as the z/OS host. So, what do we do?

# FlashCopy Overview



t <sub>0</sub>
t <sub>1</sub>
t <sub>0</sub>
t <sub>0</sub>
t <sub>0</sub>
t <sub>0</sub>
t <sub>0</sub>

1
0
1
1
1
1
1

t <sub>0</sub>

A FlashCopy is also a two-phase process

- Logical establish
- Physical data movement

The logical establish occurs when an application, like DFSMSdss, requests a set of tracks be copied from one volume A to another volume B at a point in time called  $t_0$

Once the logical establish is complete applications can resume access to the data sets on volume A while the physical data movement is performed in the background

- The physical data movement phase is performed at some later time by a process running on the storage system that copies the tracks from volume A to volume B

# FlashCopy Overview - Reads



t <sub>0</sub>
t <sub>1</sub>
t <sub>0</sub>
t <sub>0</sub>
t <sub>0</sub>
t <sub>0</sub>
t <sub>0</sub>

1
0
1
1
1
1
1

t <sub>0</sub>

When a request comes in to read data from a track on volume A

- *the storage system services the request by returning the data from the track on volume A*

When a request comes in to read data from a track on volume B that has not been copied yet

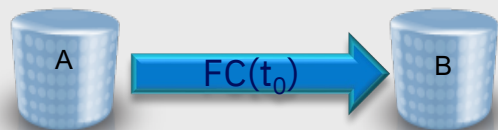
- *the storage system services the request by returning the data from the track on volume A*

When a request comes in to read data from a track on volume B that has been copied

- *the storage system services the request by returning the data from the track on volume B*



# FlashCopy Overview - Writes



$t_0$
$t_1$
$t_0$
$t_0$
$t_0$
$t_0$
$t_0$

1
0
1
1
1
1
1

$t_0$

When a request comes in to write data to a track on volume A that has not been copied yet

- *the storage system first writes the  $t_0$  copy of the track to volume B before writing the new data to volume A*

When a request comes in to write data to a track on volume A that has been copied yet

- *the storage system writes the new data to volume A*

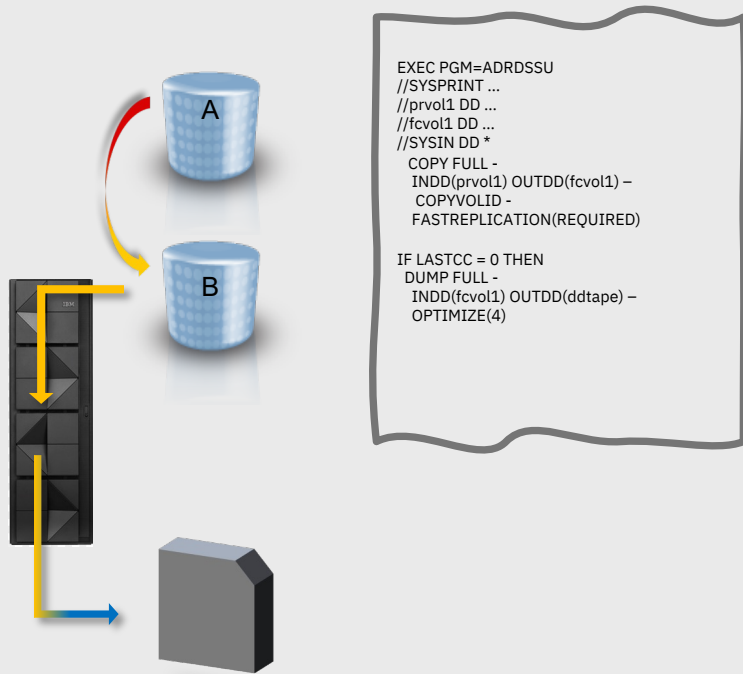
When a request comes in to write data to a track on volume B that has not been copied yet

- *the storage system writes the new data to volume B, and removes that track from the relationship table*

When a request comes in to write data to a track on volume A that has already been copied

- *the storage system writes the new data to volume B*

# Volume Dump using FlashCopy



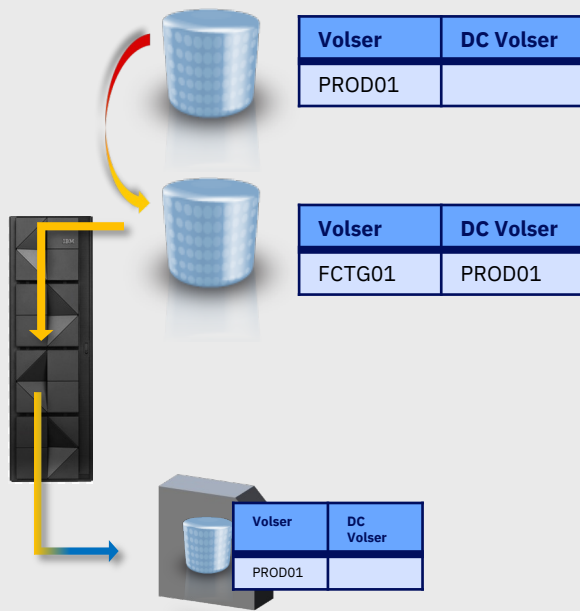
There are two cases to consider:

- DFSMSDss requests a read of data that has been copied by the storage system,
  - The read request is serviced directly from the tracks on the volume B
- DFSMSDss requests a read of data has not been copied by the storage system
  - The read request cannot get the data directly from the tracks on volume b, so the storage system services the read by getting the data from volume A

We no longer must be concerned about stress on the cache affecting production workloads but...

*We have our next challenge to overcome, duplicate volume serial numbers!*

# Volume Dump using FlashCopy – Dump Conditioning Overview



*DFSMSdss requires volumes to be online to process them and we want to dump from an exact copy of our production volumes.*

*How can we make an exact copy, volume label and all, and have both volumes online?*

The DFSMSdss COPY command has a keyword called Dump Conditioning. When the DUMPCONDITIONING keyword is specified, DFSMSdss will create a copy of the volume but will keep the volume serial number in the volume label unchanged. It will store the original volume serial number in a reserved portion of the volume label.

# Comparing the different types of FlashCopy

## Full Background Copy vs No Background Copy

- Full Background Copy is the default for DFSMSdss FlashCopy
- No Background Copy is requested using the FCNOCOPY keyword on the COPY command
  - Reads for tracks not updated are redirected to production volume ... possible impact on response time

## Full Background Copy vs Incremental Copy

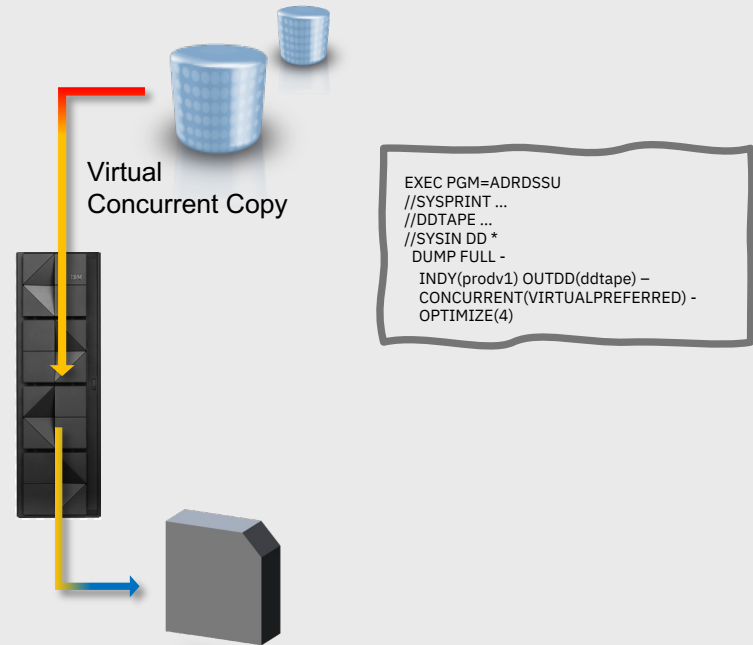
- Full Background Copy for many volumes every night may cause a lot of overhead in the storage system copying tracks from the source to the target volumes
- Incremental Copy is requested using the FCINCREMENTAL keyword on the COPY command
  - The first FlashCopy with FCINCREMENTAL is equivalent to a FlashCopy with Full Background Copy
  - Subsequent FlashCopies with FCINCREMENTAL will only copy the tracks that have changed since the previous FlashCopy
  - Reduce the amount of data that needs to be copied in the background by using incremental FlashCopy
- ***Requires dedicated FlashCopy target volumes***

# What about Virtual Concurrent Copy

Virtual Concurrent Copy is a technology that combines the concept of Concurrent Copy but uses a FlashCopy working space data set instead of cache.

The advantage of Virtual Concurrent Copy is that it does not use the cache of the storage controller for tracks that have been updated, but an additional burden is placed on you because you must properly size your working space data sets for DFSMSdss to use when FlashCopying data.

If you compare this to the COPY followed by DUMP approach, the COPY essentially takes care of creating the 'working space data sets automatically'.



# All That is great so what should I use

Most of the time we see clients use FlashCopy over Concurrent or Virtual Concurrent Copy.

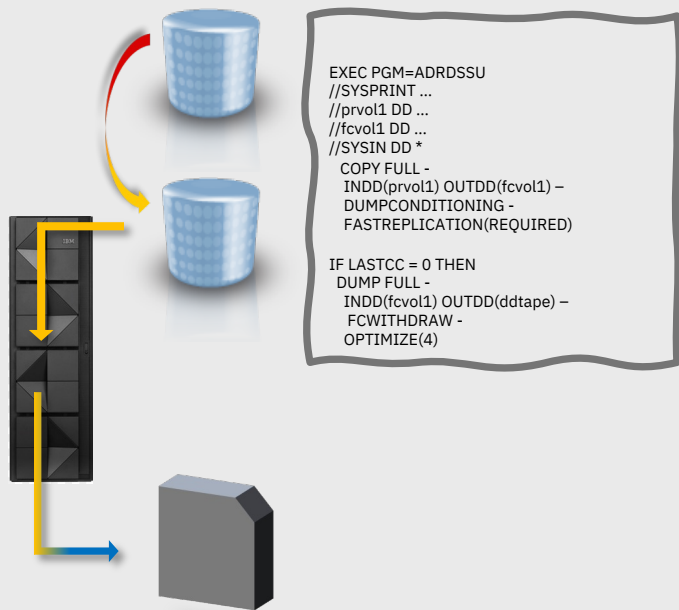
*You do manually have to add an extra COPY command before the DUMP, but other than that the benefits of FlashCopy outweigh the burden of adding the extra COPY*

Full Background Copy because their priority is to reduce the potential impact to I/O on the production volume.

If the data you are backing up is not critical application data, then NOCOPY is an acceptable approach

If you intend to backup many volumes, then you will want to consider Incremental FlashCopy to reduce the amount of background copy work the storage system has to perform

# Repeating the Dump Process Periodically



If you are using the No Background Copy (FCNOCOPY) keyword to create your FlashCopies then you must remove the FlashCopy relationship when you are done dumping the volume to tape.

The FCWITHDRAW keyword on the DUMP command will remove the FlashCopy relationship freeing up storage system resources used to track that FlashCopy relationship

DFSMSdss has made some enhancements to FCWITHDRAW support to address some of the shortcomings in the original design

# Withdraw Relationship vs ICKDSF INIT

DFSMSdss prefers to use ICKDSF INIT when FCWITHDRAW is specified in order to

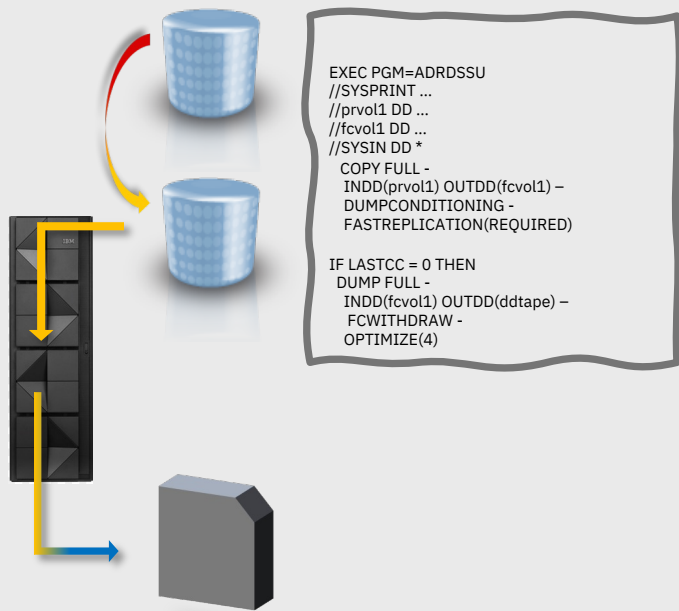
- Remove the FlashCopy relationship so that the volume can be reused for subsequent FlashCopy
- Ensure a valid VTOC on volume so that it can be reused for subsequent FlashCopy
- Remove the dump conditioned status
- Release space from the extent pool for thin provisioned volumes

Cases when DFSMSdss will not call ICKDSF, but continue with Withdraw

- VTOC already copied
- Patch byte (less common)
- ICKDSF would fail
  - Something else allocated the volume



# Repeating the Dump Process Periodically



OA53812 V2R2 and above

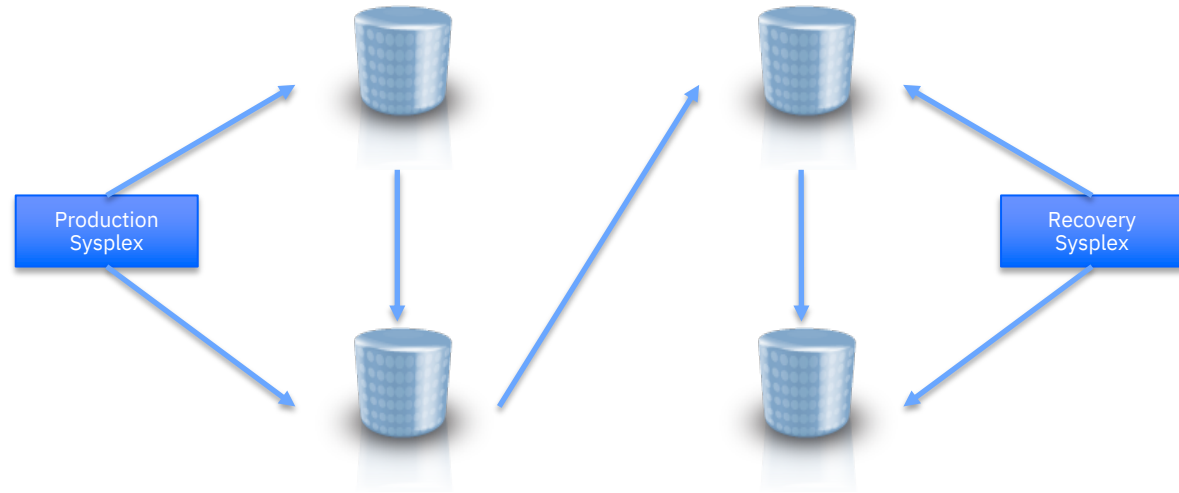
Problem:

- No warning if the volume could not be initialized
- Withdrawing when we shouldn't be
- Not always releasing space for ESE

Solution:

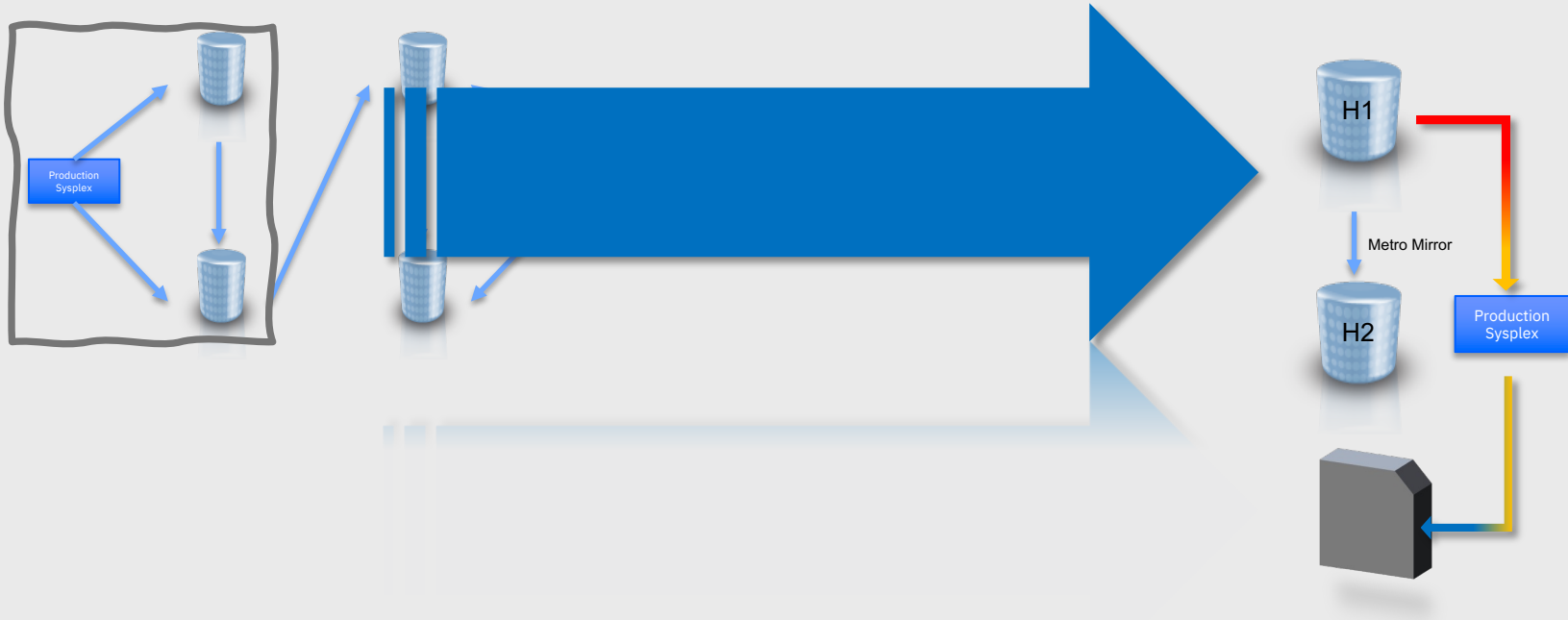
- If the volume is dump conditioned always try to INIT even if all the VTOC had been copied
- New warning message if we cannot INIT
- Not performing withdraw if we cannot INIT

# But what about my replicated volumes



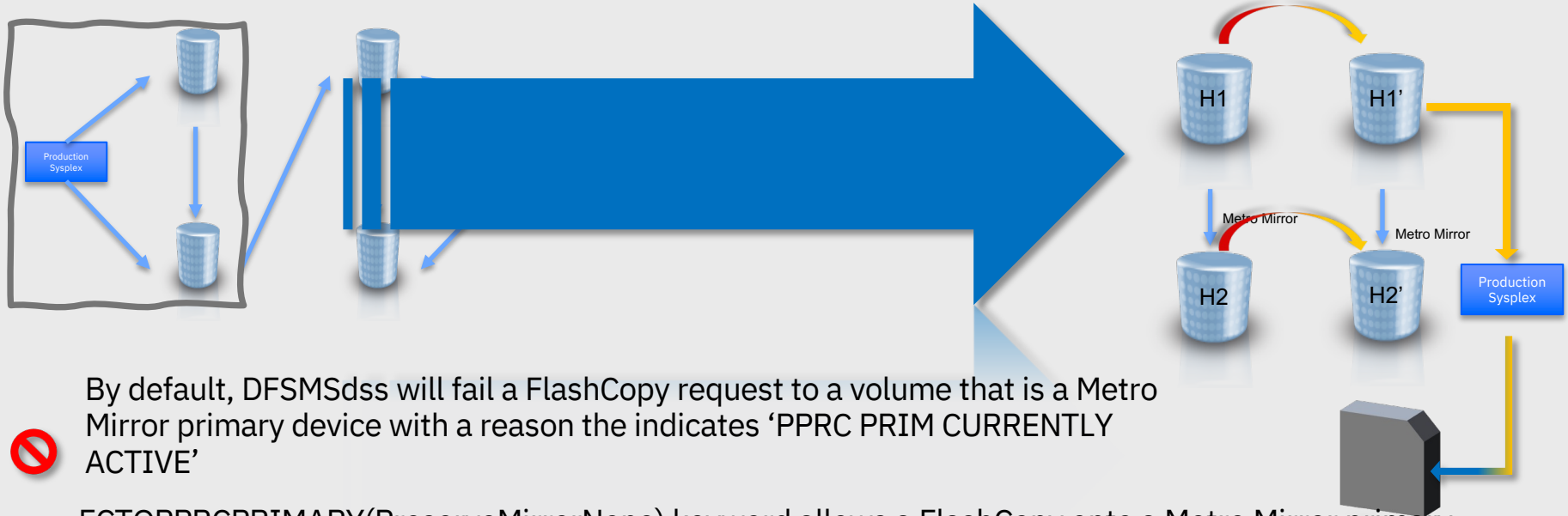
*Disk based replication technologies do an amazing job at protecting you against physical hardware failures, **but if you have a logical corruption event you now have 2 or more corrupted copies of your data.** You still need to create backups!*

# Concurrent Copy within a Metro Mirror Environment



*If a HyperSwap occurs while you are backing up H1 using Concurrent Copy, then the dump will fail, and you must restart the dump from the beginning*

# FlashCopy within a Metro Mirror Environment



By default, DFSMSdss will fail a FlashCopy request to a volume that is a Metro Mirror primary device with a reason the indicates 'PPRC PRIM CURRENTLY ACTIVE'

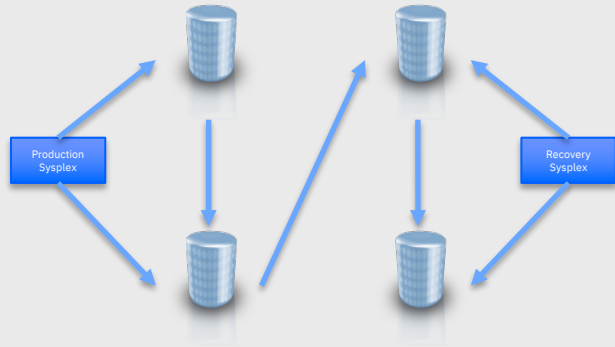


FCTOPPRCPPRIMARY(PreserveMirrorNone) keyword allows a FlashCopy onto a Metro Mirror primary volume, but the volume goes duplex pending until all the tracks are copied losing your HyperSwap capability



Specify the FCTOPPRCPPRIMARY(PreserveMirrorRequired) keyword to allow a FlashCopy onto a Metro Mirror primary volume **so that you remain HyperSwap enabled and in the event of a HyperSwap the dump process is unaffected!**

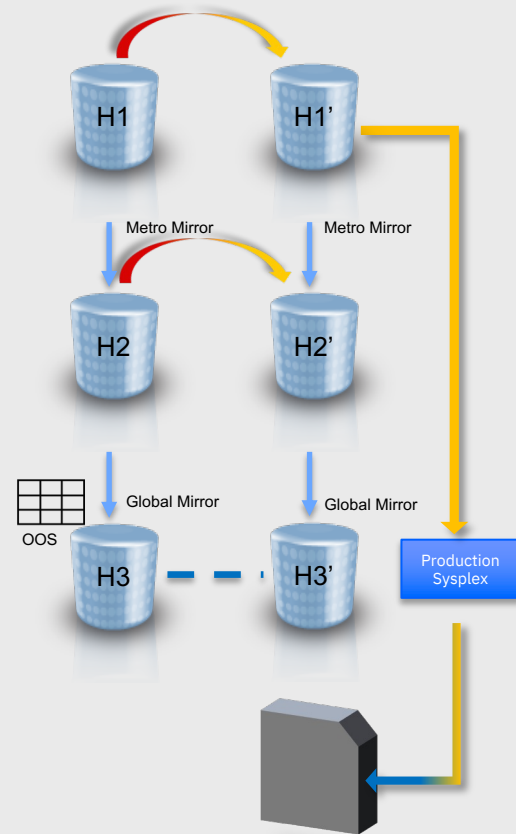
# FlashCopy within a Global Mirror (or MGM) Environment



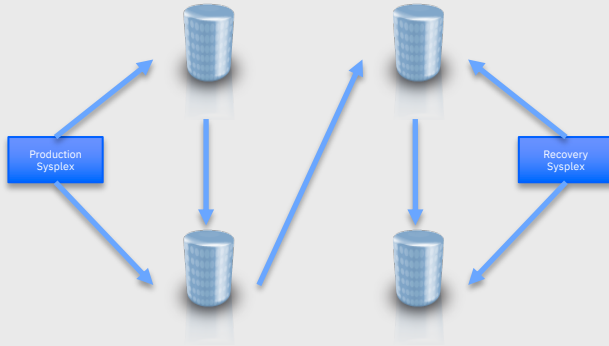
DEVSUPxx  
ENABLE(FLASHCOPYTOGM)

By default, DFSMSdss will fail a FlashCopy request to a volume that is a Global Mirror primary device with a reason the indicates 'ASYNC PPRC PRIM CURRENTLY ACTIVE'

Specify the FCTOPPRCPPRIMARY(PreserveMirrorRequired) keyword and enable FlashCopy onto Global Mirror parameter in DEVSUPxx parmlib member to allow a Global Mirror Primary become a FlashCopy target



# FlashCopy onto Global Mirror Primaries



*Since the FlashCopy command is not mirrored to the Global Mirror secondary, using the function can increase your RPO.*

*Two factors to consider are the amount of data you are FlashCopying and the frequency at which the FlashCopies are occurring.*

## Requirements

- DS8000 requirements:*
  - DS8880*  
*R8.5SP4 GA 88.54.31.0 – R18g.8b190926a*
  - DS8900*  
*R9.02 GA 89.2.38.0 – R19d.9b200310a*
- z/OS requirements*  
*z/OS V2.3, z/OS V2.4 and z/OS V2.5*  
*PTFs for OA57172, OA57173, OA57174 and OA57175*

# Shifting gears to recovery

If you use either GM or MGM then you are either not using FlashCopy for volume dumps or your FlashCopy targets are outside of your GM/MGM configuration (Potentially using Global Copy to replicate to DR region)

- *Assuming tape grid replicated to DR region the volume dump is always available for recovery from tape*

## Recovery options at production region

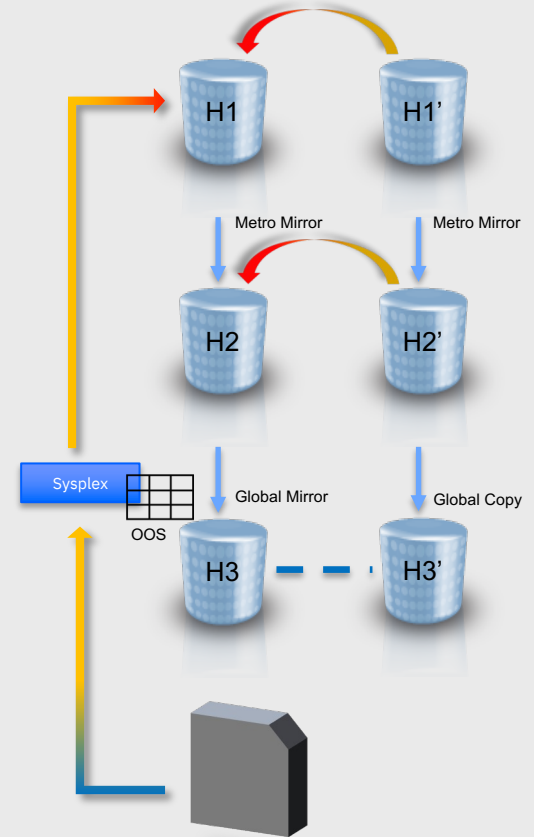
- Recovery from tape or if you have dedicated FlashCopy target volumes using either Full Background Copy or Incremental FlashCopy then you could also recover from your most recent FlashCopy

## Recovery options at disaster recovery region

- If you are not replicating your FlashCopy targets, then the only option for recovery is from tape otherwise you could use your disk copy as the most recent recovery point, you may want to enable FlashCopy onto GM because

## Recoveries from disk

- *With the support disabled even a recovery from disk is a slow copy*
- *With the support enabled a recovery from disk can now use FlashCopy*



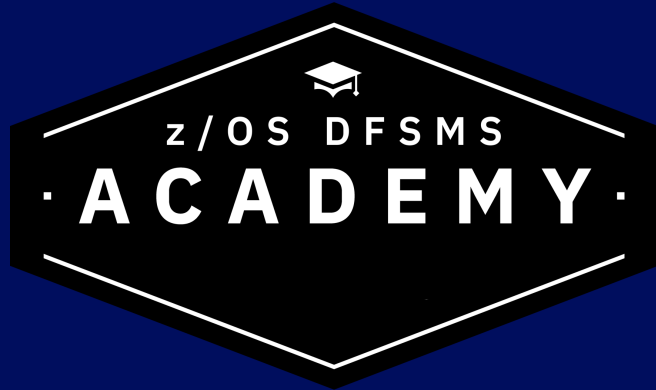
# Learning more about FlashCopy

- IBM DS8000 Copy Services Redbook
  - <https://www.redbooks.ibm.com/redpieces/abstracts/sg248367.html?Open&pdfbookmark>
- DFSMSDss Storage Administration Reference
  - <https://www.ibm.com/docs/en/zos/2.5.0?topic=dfsms-zos-dfsmsdss-storage-administration>



# DFSMSdss Deep(er) Dive Transparent Cloud Tiering

Carly Fife  
DFSMSdss Development  
[carly.fife@ibm.com](mailto:carly.fife@ibm.com)



# Transparent Cloud Tiering (TCT)

## DFSMSdss dumps and restores data using DS8000 TCT interface

- New *CLOUD*, *CONTAINER*, *OBJECTPREFIX* and *CDACREDS* keywords on *DUMP* and *RESTORE* commands

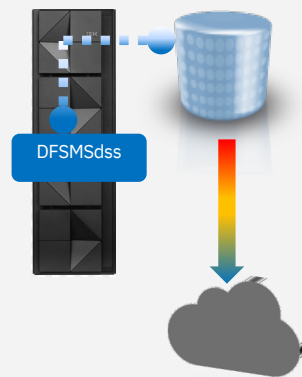
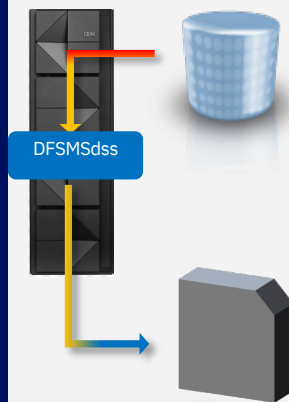
## CLOUDUTILS LIST and DELETE commands to help manage DFSMSdss dumps

- *CLOUDUTILS LIST* to lookup information about dumps in a cloud
- *CLOUDUTILS DELETE* to delete dumps in a cloud

### Notes:

- Volumes must be online to be used by dss
- Physical data set restore is not supported for volumes dumped using TCT

*Transparent Cloud Tiering completes the evolution of serverless data movement through your entire storage hierarchy!*



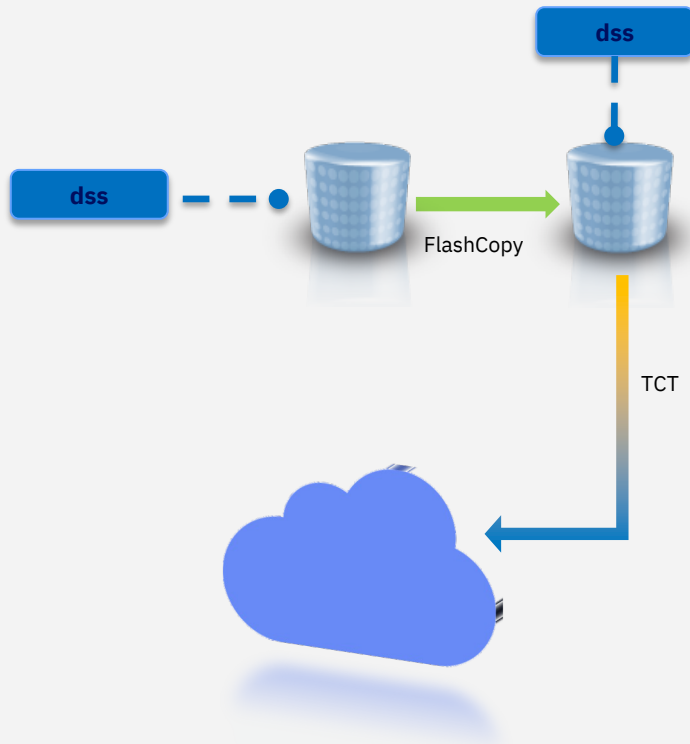
# Full volume dumps from FlashCopy targets

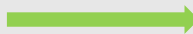

## Steps

- *FlashCopy volume with DUMPCONDITONING*
- *DUMP volume using TCT*

## Notes

- *Supports regular FlashCopy with full background copy, incremental FlashCopy as well as NoCopy relationships*
- *You can also use existing FlashCopy targets taken outside of dss, just condition the volumes using ICKDSF REFORMAT DUMPCOND(SET) and bring them online*
- *Dumps can be taken from a primary replication site or from a secondary replication site*



Legend	
	FlashCopy
	TCT

# Extending FlashCopy cyber resilience capabilities

**Volume dumps from global mirror secondary provide crash consistent backups of your data**

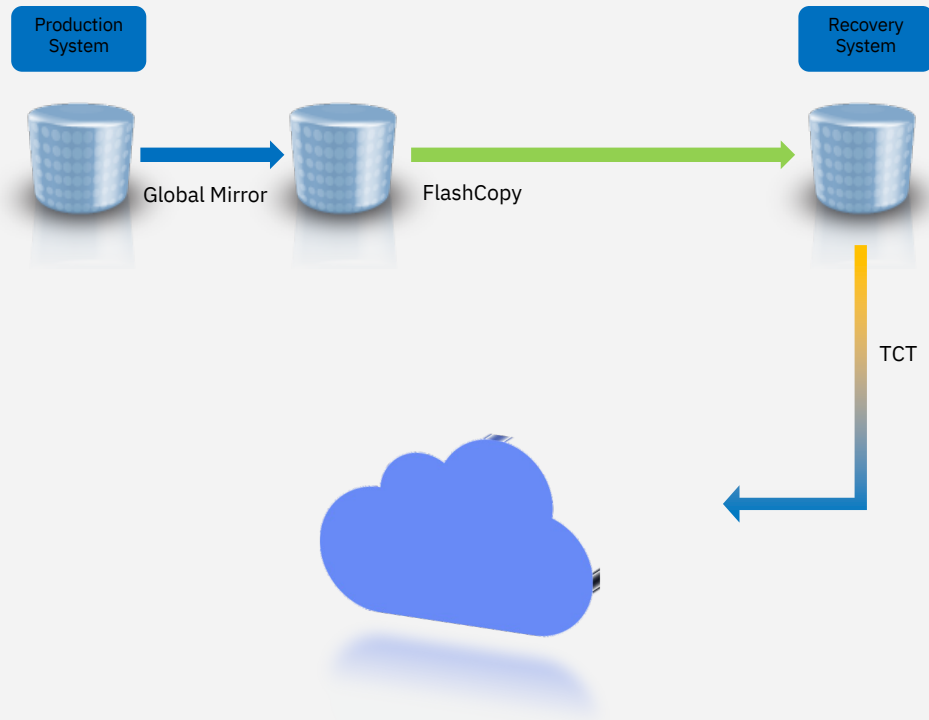
- *Pause global mirror*
- *FlashCopy volumes with DUMPCONDITONING*
- *Dump volumes to cloud using TCT*

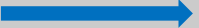
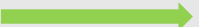

**DFSMSdss may see VTOC/VTOCIX in the middle of an update**

- *Cannot trust free space information and dump all the unallocated space along with the allocated space (ADR310W and RC=4)*

## Notes

- *Install PTFs for APAR OA58758 and patch dss to convert ADR310W to ADR529I and RC=0*



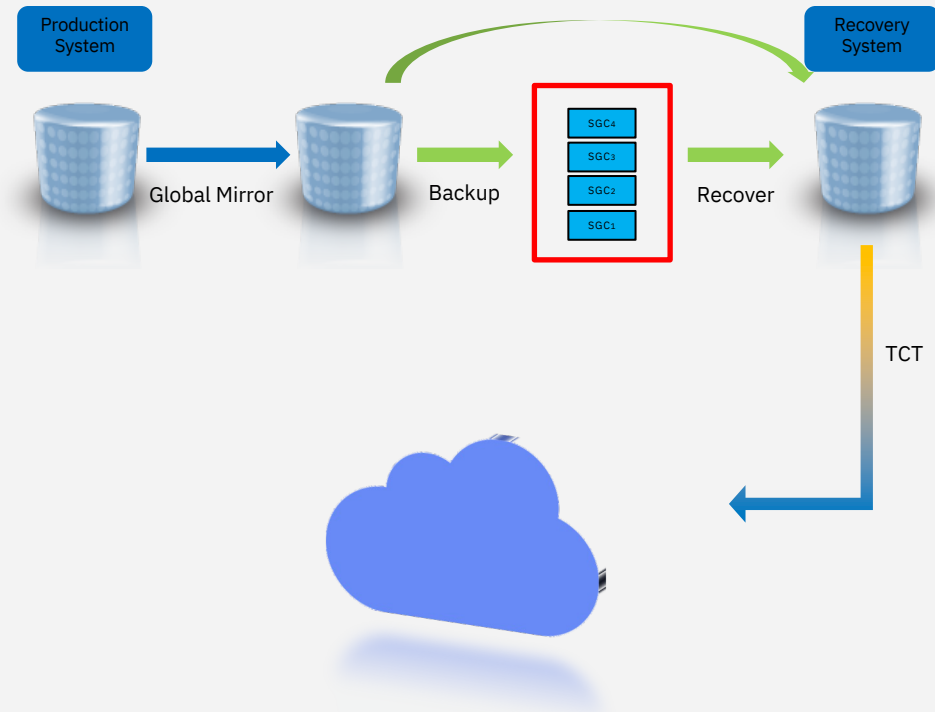
Legend	
	Mirroring
	FlashCopy
	TCT

# Extending safeguarded copy cyber resilience capabilities

**Safeguarded copies are designed to help protect you from logical corruption events, but the number of safeguarded copies that can be kept online is finite**

**Periodically recover safeguarded copies to a recovery system, perform validation and then dump to cloud using dss full volume dump**

- *Use logical corruption protection manager like GDPS or CSM to perform a safeguarded recovery*
- *Dump volumes to cloud using DFSMSdss and TCT*

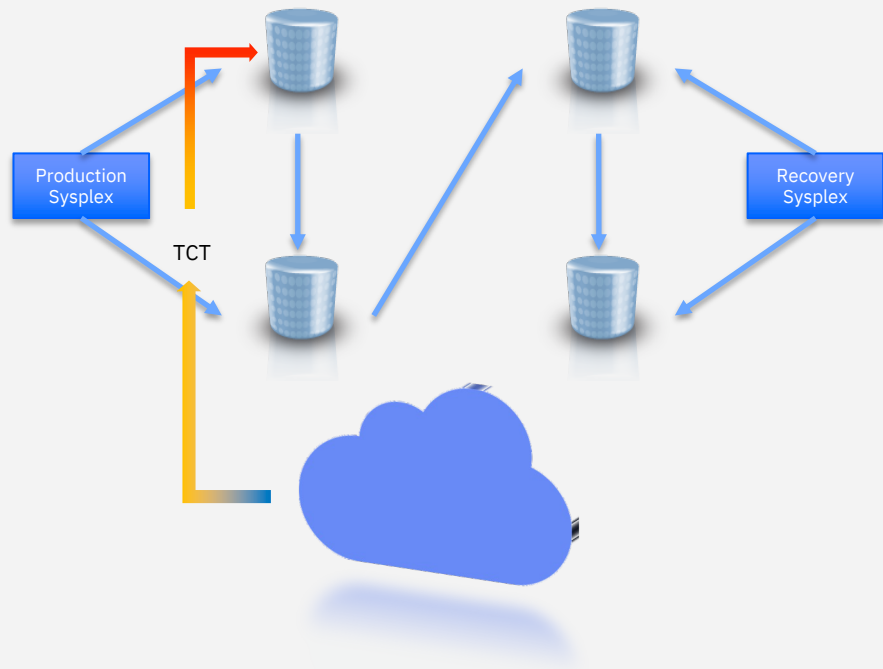


Legend	
	Mirroring
	Safeguarded Copy
	TCT

# Restoring volumes back into production

Whether the data was dumped directly from production, a FlashCopy target or a Safeguarded Recovery volume, the restore can be performed directly back into production

```
//RESTORE EXEC PGM=ADRDSSU
//DISK1 DD UNIT=SYSDA,DISP=SHR,VOL=SER=VOL001
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
RESTORE -
  FULL COPYVOLID -
  OUTDDNAME ( DISK1 ) -
  CLOUD( TS7700CLOUD ) -
  CDACREDS -
  CONTAINER( DSSDUMPS ) -
  OBJECTPREFIX( 'DNIGHTLY/VOL001/D20220819' )
```



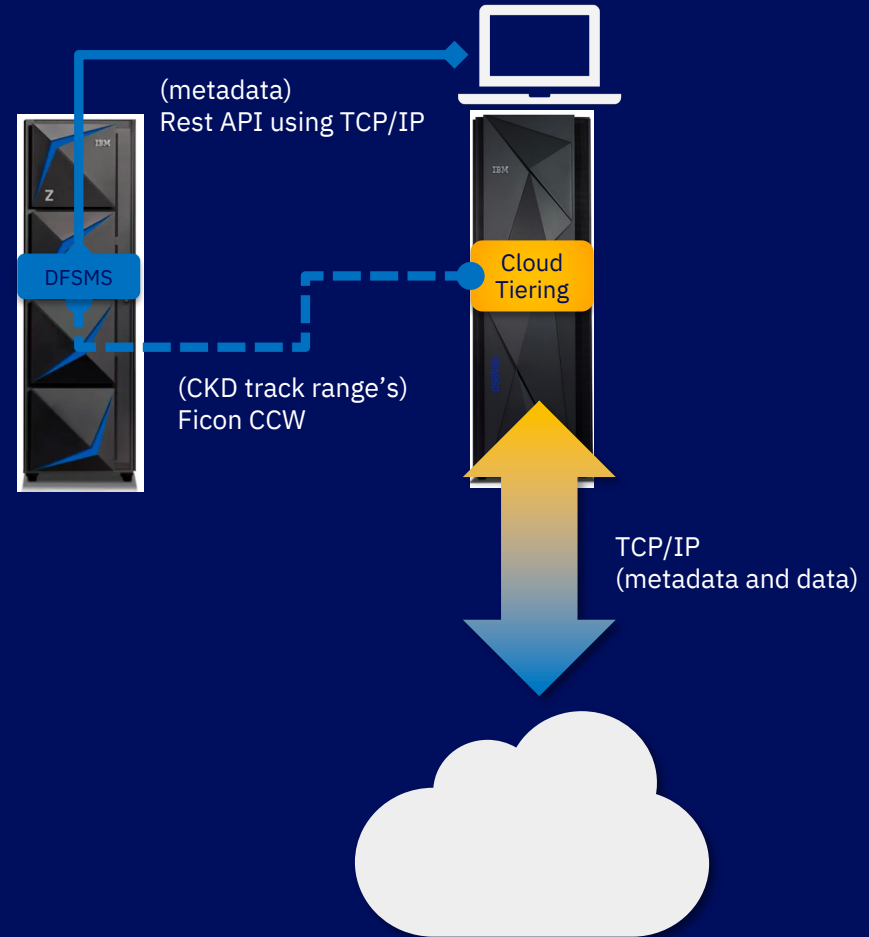
**Physical data set restore is not supported for volumes dumped using TCT**

# Implementing and Managing DFSMSdss Full Volume Dumps using TCT



# Data Flow

- **DFSMS stores both metadata and data**
  - CKD track data written over ethernet by DS8000
    - FICON CCW tells DS8000 which CKD tracks to store
  - DFSMS uses DS8000 HMC to write metadata and we must authenticate with the HMC with a username and password
    - Endpoint and username is in the SMS network connection, password is managed in a couple different ways by the applications using TCT





# Credentials Management in DFSMSdss using CDA

```
//DUMP EXEC PGM=ADRDSSU
//PRODVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=&PRDVOL
//FLCVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=&FLCTGT
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
... COPY with FlashCopy ...

IF LASTCC = 0 THEN
  DUMP FULL INDDNAME(FLCVOL) FCWITHDRAW -
    ALLDATA ALLEXCP(*) -
    CCREDS(password) -
    CLOUD( &CLOUD ) -
    CONTAINER( &CONTNER ) -
    OBJECTPREFIX( ' &JOBNAME/&PRDVOL/D&STODAY' )
```

```
//DUMP EXEC PGM=ADRDSSU
//PRODVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=&PRDVOL
//FLCVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=&FLCTGT
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
... COPY with FlashCopy ...

IF LASTCC = 0 THEN
  DUMP FULL INDDNAME(FLCVOL) FCWITHDRAW -
    ALLDATA ALLEXCP(*) -
    CDACREDS -
    CLOUD( &CLOUD ) -
    CONTAINER( &CONTNER ) -
    OBJECTPREFIX( ' &JOBNAME/&PRDVOL/D&STODAY' )
```

*No longer risk exposing password in JCL*

# Securing credentials using CDA (GDKAUTHP)

From ISPF exec 'SYS1.SAXREXEC(GDKAUTHP)'

```
z/OS Cloud Data Access Authorization Utility
Option ==> S                                Key saved/updated...
      S Save Resource Authorization          C Clear Secret Key Field
                                           (for hidden input)

Encryption Parameters
Provider . . . TS7700CLOUD
KeyLabel . . . GDK.DSSADMIN.TS7700CLOUD
Keystore . . . /u/dssadmin/gdk/gdkkeyf.json
Resource . . . /

Authorization Parameters
Key . . . . userid
Secret Key . . *****
```

contents of config.json

```
{
  ...
  "allow-no-CEX": true,
  ...
}
```

*CDA will encrypt the username and password with a key label it creates, and the key is stored in ICSF  
By default, it will 'wrap' the key used to encrypt the credentials using the master key of a Crypto Express adapter*

*ERROR: encryptKeys: Unable to generate a key. CSNBKGN rc: 12, rsn:0000*

*If you see this error, you can install PTFs for APAR OA60481 (PTFs are available on z/OS V2R3 and above)*

*If the system does not have any Crypto Express configured as CCA co-processor and allow-no-CEX is set to true, then CDA will store the key in ICSF in the clear*

*Hint: If you don't have CEX adapters and set allow-no-CEX to true but still see this CSNBKGN rc: 12, rsn:0000*

*Check the format of your CKDS it could be using fixed length record format (LRECL=252) check your CKDS with LISTCAT*

# Inventory Management

## *CLLOUDUTILS LIST*

- Lists one ore more dumps in a container or all the dumps in a container*
- Access to the CLOUDUTILS LIST command requires at least read access to STGADMIN.ADR.CLOUDUTILS facility class profile*



# CLOUDUTILS LIST

*-CLOUDUTILS LIST CDACREDS CLOUD(TS7700CLOUD) CONTAINER(DSSDUMPS)*

0ADR605I (001)-CUTIL(03), THE FOLLOWING DUMP DATA SETS WERE FOUND  
+ DNIGHTLY/VOL001/D20220813  
...  
+ DNIGHTLY/VOL001/D20220819  
+ DNIGHTLY/VOL002/D20220813  
..  
+ DNIGHTLY/VOL002/D20220819

*-CLOUDUTILS LIST CDACREDS CLOUD(TS7700CLOUD) CONTAINER(DSSDUMPS) OBJECTPREFIX('DNIGHTLY/VOL001')*

0ADR605I (001)-CUTIL(03), THE FOLLOWING DUMP DATA SETS WERE FOUND  
+ DNIGHTLY/VOL001/D20220813  
...  
+ DNIGHTLY/VOL001/D20220819

# Wildcards/LIST ALL

## WILDCARDS

- Limited wildcard support for OBJECTPREFIX keyword
- Wildcard must be at the end of the phrase
- CONTAINER does not support wildcards

## LIST ALL

- Each dss dump is made up of multiple objects placed in the cloud
- By default, LIST creates a list of dumps
- LIST ALL lists the objects in the cloud separately.

# Object Expiration



## CLOUDUTILS DELETE

- Deletes one or more dumps in a container or deletes an entire container
- Access to the CLOUDUTILS DELETE command requires at least read access to STGADMIN.ADR.CLOUDUTILS.DELETE facility class profile
- Deleting non-empty containers with CLOUDUTILS DELETE FORCE requires at least read access to STGADMIN.ADR.CLOUDUTILS.FORCE facility class profile

# CLOUDUTILS DELETE

```
-CLOUDUTILS DELETE CDACREDS CLOUD(TS7700CLOUD) CONTAINER(DSSDUMPS)
```

0ADR610E (001)-CUTIL(02), 2020.279 13:28:57 AN UNEXPECTED ERROR HAS BEEN ENCOUNTERED, 4

```
-CLOUDUTILS DELETE CDACREDS CLOUD(TS7700CLOUD) CONTAINER(DSSDUMPS) FORCE
```

0ADR605I (001)-CUTIL(03), THE FOLLOWING DUMP DATA SETS WERE DELETED  
+ DNIGHTLY/VOL001/D20220813  
...  
+ DNIGHTLY/VOL001/D20220819  
+ DNIGHTLY/VOL002/D20220813  
...  
+ DNIGHTLY/VOL002/D20220819

```
-CLOUDUTILS DELETE CDACREDS CLOUD(TS7700CLOUD) CONTAINER(DSSDUMPS) OBJECTPREFIX('DNIGHTLY/VOL001')
```

0ADR605I (001)-CUTIL(03), THE FOLLOWING DUMP DATA SETS WERE DELETED  
+ DNIGHTLY/VOL001/D20220813  
...  
+ DNIGHTLY/VOL001/D20220818

# Managing only DFSMSdss Containers



- When invoked from batch JCL, DFSMSdss will add a prefix to the container name and will only list or delete containers with the designated prefix
- When invoked from an application, DFSMSdss will not modify the container name passed as input

```
DUMP INDDNAME(DISK1) CLOUD(cloudname) CCREDS(password) -  
CONTAINER(DSSDUMPS) OBJECTPREFIX(objectprefix1)-  
WAIT(0,0) ALLDATA(*) ALLEXCP
```



# Putting it all together using TCT for full volume dumps



*Suppose you have some existing JCL to create nightly dumps of a volume to tape and keep one weeks worth of dumps.*

*We will walk through the process of modifying the JCL to start directing those dumps to object storage using TCT*

*One challenge to overcome is handling the expiration of old dumps as new ones are created*

*We will do this with template JCL, REXX and the file tailoring service*

# DFSMSdss JCL Skeleton

```
//DNIGHTLY JOB...
//DUMP EXEC PGM=ADRDSSU
//PRODVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=&PRDVOL
//FLCVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=&FLCTGT
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
COPY FULL INDDNAME(PRD001) OUTDDNAME(FLCVOL) -
      FCNOCOPY FCTOPPRCP(PresMirReq) DUMPCONDITIONING

IF LASTCC = 0 THEN
  DUMP FULL INDDNAME(FLCVOL) FCWITHDRAW -
        ALLDATA ALLEXCP(*) -
        CDACREDS CLOUD( &CLOUD ) CONTAINER( &CONTNER ) -
        OBJECTPREFIX( ' &JOBNAME/&PRDVOL/D&STODAY' )

IF LASTCC = 0 THEN
  CLOUDUTILS DELETE -
        CDACREDS CLOUD( &CLOUD ) CONTAINER( &CONTNER ) -
        OBJECTPREFIX( ' &JOBNAME/&PRDVOL/D&SEXPIRE' )
/*
```

*It is up to you to choose an object prefix naming convention*

*Use symbols to create an object prefix naming convention that will allow for multiple backups, and multiple versions of a backup, in the same container*

*Using a date in the object prefix is one way to allow for multiple versions of the same backup*

*Note for the sysprog: Skeleton's must be in a library that is part of the ISPF.ISPSLIB concatenation  
CONCATD F(ISPSLIB) DA(my.skels) SHR BEFORE*

# File Tailoring

*We need a way to delete old dumps based on our retention policy*

*We need to do some arithmetic to find the dump with the object prefix that is to be deleted*

*Sample REXX provided to calculate an expiration date and set other symbols required in the DUMP JCL*

```
/* REXX */
/*****
/* https://www.rexxla.org/rexxlang/mfc/datec.html */
/*
/* Also, credit to expat from
/* http://ibmmainframes.com/about66336.html */
*****/

numVersions = 7
cloud = TS7700CLOUD
contner = DSSDUMPS
prdv01 = VOL001
flctgt = FLC001

/* get today's date in 'Base' date format, e.g. '729584' */
today = Date('Base')

/* calculate date that we want to expire; returns '729577' */
expire = today - numVersions

stoday = Date('Standard', today, 'Base')
sexpire= Date('Standard', expire, 'Base')

"free fi(ispfile)"
"del  '"uid".ispfile'"
"alloc fi(ispfile) da('"uid".ispfile') new tracks space(5 5)
      recfm(f b) lrecl(80)"

/* EDIT The tailored file and submit manually */
address ispxexec "ftopen"
address ispxexec "ftincl nightly"
address ispxexec "ftclose"
address ispxexec "edit dataset('"uid".ispfile')"

EXIT
```

# Real World Use Case



```
//DNIGHTLY JOB...
//DUMP EXEC PGM=ADRDSSU
//PRODVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=VOL001
//FLCVOL DD UNIT=SYSDA,DISP=SHR,VOL=SER=FLC001
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
COPY FULL INDDNAME(PRD001) OUTDDNAME(FLCVOL) -
FCNOCOPY FCTOPPRCP(PresMirReq) DUMPCONDITIONING

IF LASTCC = 0 THEN
DUMP FULL INDDNAME(FLCVOL) FCWITHDRAW -
ALLDATA ALLEXCP(*) -
CDACREDS CLOUD( TS7700CLOUD ) CONTAINER( DSSDUMPS ) -
OBJECTPREFIX( 'DNIGHTLY/VOL001/D20220819' )

IF LASTCC = 0 THEN
CLOUDUTILS DELETE -
CDACREDS CLOUD( TS7700CLOUD ) CONTAINER( DSSDUMPS ) -
OBJECTPREFIX( 'DNIGHTLY/VOL001/D20220812' )

/*
```

# Restoring a full volume dump

- ***Given a volume to be restored, use the CLOUDUTILS LIST Command to display the list of backups for that volume***

```
- CLOUDUTILS LIST CDACREDS CLOUD(TS7700CLOUD) CONTAINER(DSSDUMPS) OBJECTPREFIX('DNIGHTLY/VOL001')
0ADR605I (001)-CUTIL(02), 2020.279 13:33:21 THE FOLLOWING DUMP DATA SETS WERE FOUND
+
+          DNIGHTLY/VOL001/D20220813
+          DNIGHTLY/VOL001/D20220814
+          DNIGHTLY/VOL001/D20220815
+          DNIGHTLY/VOL001/D20220816
+          DNIGHTLY/VOL001/D20220817
+          DNIGHTLY/VOL001/D20220818
+          DNIGHTLY/VOL001/D20220819
```

- ***Update the RESTORE JCL to specify the cloud name, container name and the object prefix to be restored***

```
//RESTORE EXEC PGM=ADRDSSU
//DISK1 DD UNIT=SYSDA,DISP=SHR,VOL=SER=VOL001
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
RESTORE -
FULL COPYVOLID -
OUTDDNAME ( DISK1 ) -
CLOUD( TS7700CLOUD ) -
CDACREDS -
CONTAINER( DSSDUMPS ) -
OBJECTPREFIX( 'DNIGHTLY/VOL001/D20220818' )
```

# DFSMSdss DUMP/RESTORE FULL Requirements

*Native Full Volume Dump and Restore is supported on z/OS V2R3 and z/OS V2R4 with PTFs for APAR OA57526*

- *Includes support for CDACREDS and CLOUDUTILS LIST and DELETE capabilities*
- *Check for updates using the IBM.Function.DFSMSCloudStorage (DFSMSCS/K) fix category*

*Support using the DS8000 Object Store Feature of the TS7700 as well as traditional object stores*

- *TS7700 as an object store is supported on VEC/VED model with 8.50.x code or above and requires FC5282 – DS8000 Object Store*

*Supported on both DS8880 and DS8900F with microcode upgrades*

- *DS8880 Release 8.5 SP6*
- *DS8900F Release 9.1*

# Learning more about Transparent Cloud Tiering

DS8900F Transparent Cloud Tiering Redbook

<https://www.redbooks.ibm.com/abstracts/sg248381.html>

DFSMSdss Storage Administration Reference

<https://www.ibm.com/docs/en/zos/2.5.0?topic=dfsms-zos-dfsmsdss-storage-administration>

Consolidated PDF with TCT updates to z/OS Manuals

<http://publibz.boulder.ibm.com/zoslib/pdf/OA51622.pdf>

# Thank you

Ernesto Figueroa  
DFSMSdss Product Owner  
erfiguer@us.ibm.com

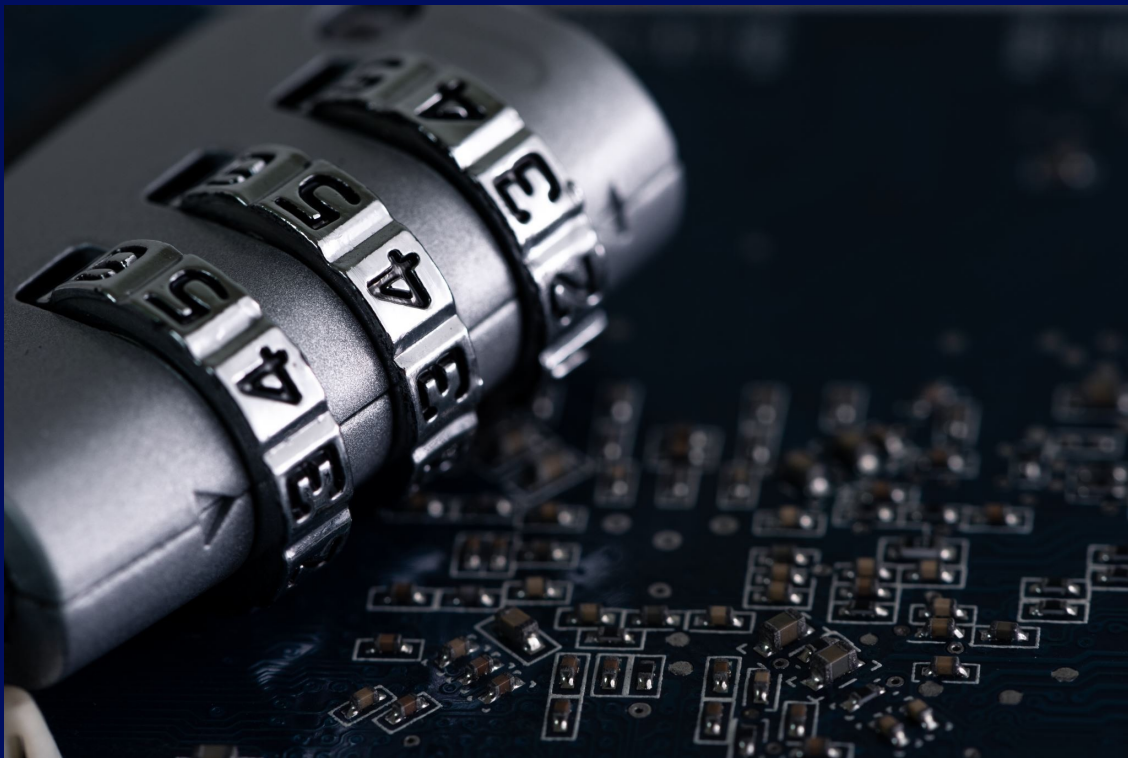
Carly Fife  
DFSMSdss Development  
Carly.Fife@ibm.com

Robert Gensler  
STSM – DFSMSHsm Product Owner  
rgensle@us.ibm.com



# Securing account credentials

- CDA Setup
- Copy CDA Configuration Samples
- Configure CSFKEYS general resource class
- Add Cloud Provider Keys



# Cloud Data Access Setup

## Assumptions

PTF for OA59933 is installed

## TODO

Ensure that SYS1.DFQPLIB is part of the ISPPLIB concatenation or that the following members located in SYS1.DFQPLIB are added to an ISPPLIB library:

GDKAPPOP  
GDKAUTHK  
GDKAUTHL  
GDKAUTHP  
GDKMAINP  
GDKOBJAC  
GDKOBJAL  
GDKOPPOP

*A RACF (or equivalent) profile should be created to ensure only authorized users have access to these members.*

# Configure CSFKEYs General Resource Class

## Assumptions

*The CSFKEYS general resource class is active and RACLISTed.*

*The CSFSERV general resource class is active and RACLISTed.*

## TODO

```
PERMIT CSFKGN* CLASS(CSFSERV) ID(dssadmin) ACCESS(READ)
```

```
PERMIT CSFKRC* CLASS(CSFSERV) ID(dssadmin) ACCESS(READ)
```

```
RDEFINE CSFKEYS GDK.DSSADMIN.* UACC(NONE)
```

```
PERMIT GDK.STGADMIN.* CLASS(CSFKEYS) ID(dssadmin) ACCESS(READ)
```

```
RALTER CSFKEYS GDK.DSSADMIN.* ICSF(SYMCPACFWRAP(YES))
```

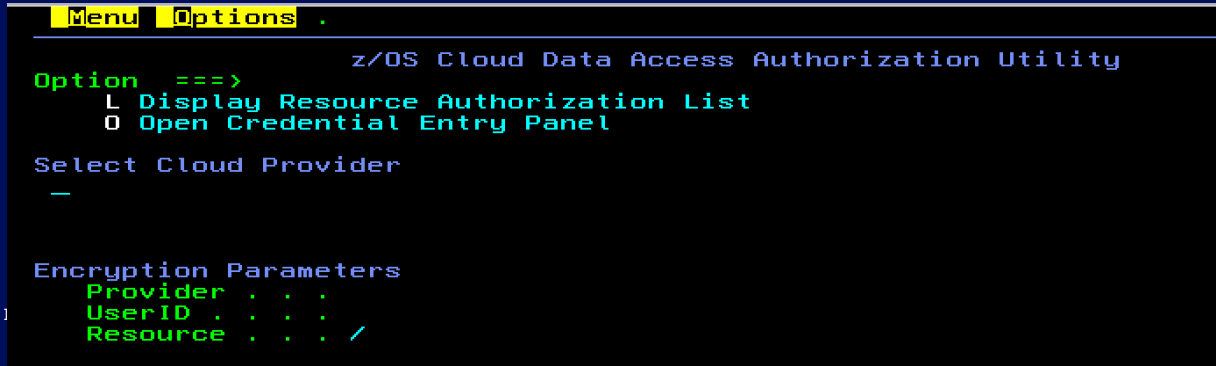
*The ICSF segment of the CSFKEYS class profiles covering the keys created by CDA must contain SYMCPACFWRAP(YES).*

## From OMVS home directory:

1. Create a gdk directory and copy setup files
2. Create a 'providers' directory and an empty file that corresponds to the cloud network connection name defined in the SMS configuration

```
$ mkdir gdk; cd gdk; pwd
/u/dssadmin/gdk
$ cp /usr/lpp/dfsms/gdk/samples/gdkconfig.json config.json
$ cp /usr/lpp/dfsms/gdk/samples/gdkkeyf.json .
```

```
$ mkdir providers
$ ls
config.json  gdkkeyf.json providers
$ touch providers/MYCLOUD.json
$ ls providers
MYCLOUD.json
```



# z/OS Cloud Data Access Authorization Utility

-----  
ISPF Primary Option Menu

20.275

Option ==> tso exec 'sys1.saxrexec(gdkauthp)'

0	Settings	Terminal and user parameters
1	View	Display source data or listings
2	Edit	Create or change source data
3	Utilities	Perform utility functions
4	Foreground	Interactive language processing
5	Batch	Submit job for language processing
6	Command	Enter TSO or Workstation commands
7	Dialog Test	Perform dialog testing
9	IBM Products	IBM program development products
10	SCLM	SW Configuration Library Manager
11	Workplace	ISPF Object/Action Workplace
12	z/OS System	z/OS system programmer applications
13	z/OS User	z/OS user applications
L	SSD-SP	SSD SP Local Selection Panel

< Calendar >

October 2020

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Time . . . . : 06:25

Day of year. : 275

Enter X to Terminate using log/list defaults

# Securing credentials using CDA (GDKAUTHP)

From ISPF exec 'SYS1.SAXREXEC(GDKAUTHP)'

```
z/OS Cloud Data Access Authorization Utility
Option ==> S                                Key saved/updated...
      S Save Resource Authorization          C Clear Secret Key Field
                                           (for hidden input)

Encryption Parameters
Provider . . . TS7700CLOUD
KeyLabel . . . GDK.DSSADMIN.TS7700CLOUD
Keystore . . . /u/dssadmin/gdk/gdkkeyf.json
Resource . . . /

Authorization Parameters
Key . . . . . userid
Secret Key . . *****
```

contents of config.json

```
{
  ...
  "allow-no-CEX": true,
  ...
}
```

*CDA will encrypt the username and password with a key label it creates, and the key is stored in ICSF  
By default, it will 'wrap' the key used to encrypt the credentials using the master key of a Crypto Express adapter*

*With PTFs for APAR OA60481, using CDACREDs no longer requires a Crypto Express adapter  
PTFs are available on z/OS V2R3 and above*

*If the system does not have any Crypto Express configured as CCA co-processor and allow-no-CEX is set to true, then CDA will store the key in ICSF in the clear*