z/OS 3.1 IBM Education Assistant

Solution Name: Externalize Cloud Data Access (CDA) APIs / Cloud Data Access Utility

Solution Element(s): DFSMSdfp CDA (Cloud Data Access)

July 2023

Agenda

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
    - None.

Objectives

- Describe the DFSMSdfp CDA Cloud Object Utility (GDKUTIL)
- Describe the DFSMSdfp CDA Application Programming Interfaces (APIs) available in z/OS V3R1

Overview – ZRM-3341 Cloud Object Utility

- Who (Audience)
    - z/OS Application Programmers that want to incorporate Cloud Object data into their JCL batch processing.
- What (Solution)
    - A new DFSMS utility, GDKUTIL, is introduced. It allows for Download of Cloud Objects to z/OS UNIX files or sequential data sets. It additionally allows for Upload of a z/OS UNIX file or sequential data set to an object in Cloud Object Storage.
- Wow (Benefit / Value, Need Addressed)
    - Previous processes that used Network Attached Storage to transfer data between z/OS and distributed systems is not secure enough. Cloud Object Storage provides a secure, auditable environment for this data.
    - The Cloud Object Utility, GDKUTIL, provides a cloud provider agnostic JCL interface allowing z/OS Batch jobs to utilize Cloud Object storage, while maintaining the secure access to the data.

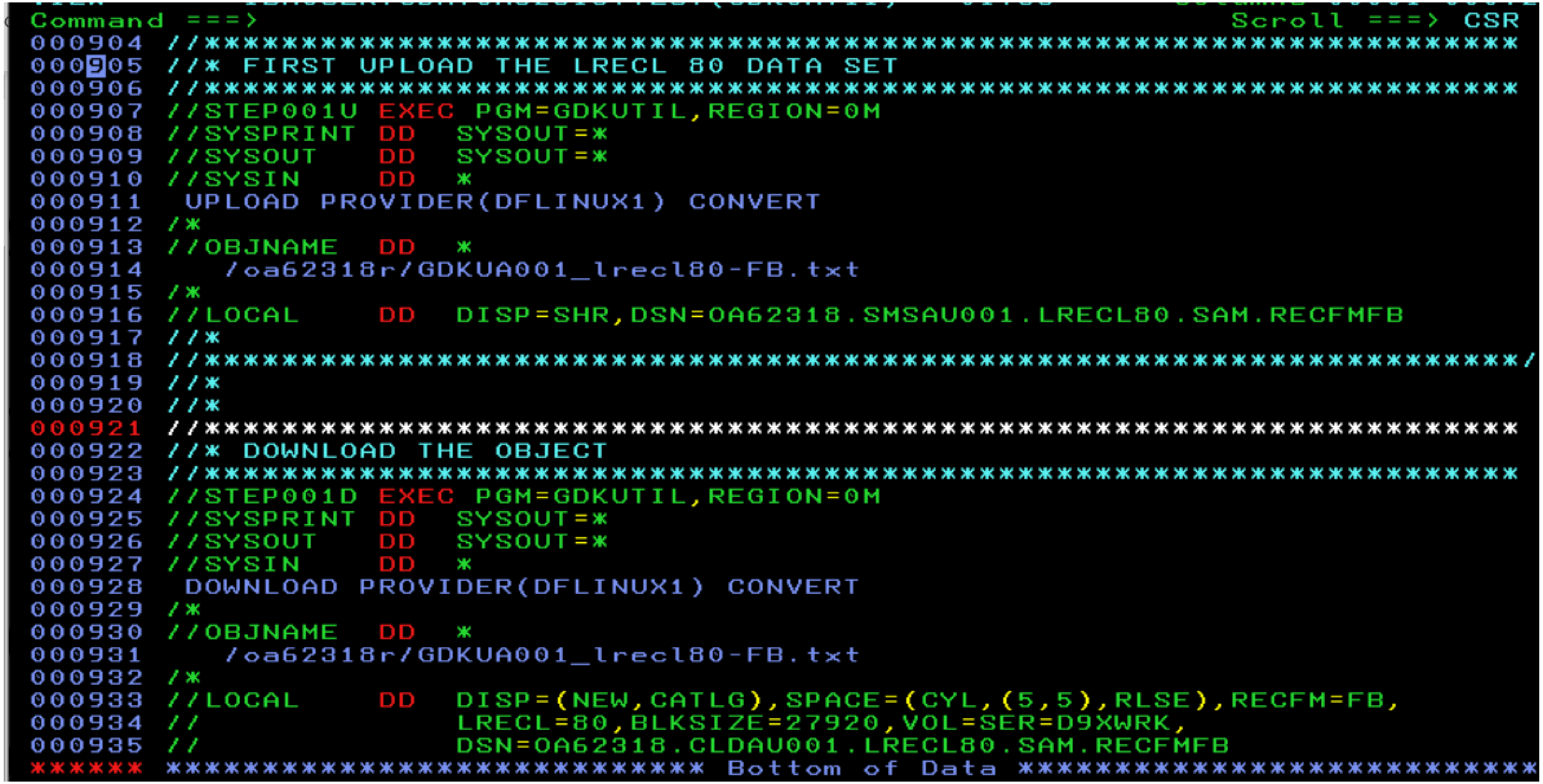
Usage & Invocation

- GDKUTIL is available in SYS1.SIEALNKE.
- System Configuration to use the CDA services invoked by GDKUTIL can be found in the z/OS MVS Programming: Callable Services for High-Level Languages in a new chapter titled, "Cloud Data Access Services"
  - System Administrator Configuration Quick-Start
  - User Configuration Quick-Start
    - Saving Cloud Credentials using the CDA Authorization Panel – SYS1.SAXREXEC(GDKAUTHP)
- Required DD names:
  - SYSPRINT – Utility messages
  - SYSIN – Command and Keyword input
  - SYSOUT – Utility logging messages
  - OBJNAME – Cloud Object name
  - LOCAL/LOCNAME – Local z/OS data.

Usage & Invocation (cont)

- SYSIN
  - Commands – UPLOAD, DOWNLOAD, LIST, DELETE
  - Keywords – PROVIDER(name), CONVERT, DELSRC, LOG(level), WEBTOOLKITLOG
- OBJNAME
  - Cloud bucket name for LIST
  - Cloud Object name for UPLOAD, DOWNLOAD, DELETE
- LOCAL – DD statement that describes a z/OS UNIX file or z/OS sequential data set that is used as the source (UPLOAD) or target (DOWNLOAD) of the command
- LOCNAME – DD statement that describes the name of the local thing to be used in the command.
  - /u/user1/dir1/file.txt
  - HLQ.DSNAME.LLQ
  - //’FULLY.QUALIFIED.NAME’

Usage example



```
Command ===>                                                      Scroll ===> CSR
//********************************************************************
//* FIRST UPLOAD THE LRECL 80 DATA SET
//********************************************************************
//STEP001U EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   *
 UPLOAD PROVIDER(DFLINUX1) CONVERT
/*
//OBJNAME  DD   *
   /oa62318r/GDKUA001_lrecl80-FB.txt
/*
//LOCAL    DD   DISP=SHR,DSN=OA62318.SMSAU001.LRECL80.SAM.RECFMFB
//*
//********************************************************************/
//*
//*
//********************************************************************
//* DOWNLOAD THE OBJECT
//********************************************************************
//STEP001D EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   *
 DOWNLOAD PROVIDER(DFLINUX1) CONVERT
/*
//OBJNAME  DD   *
   /oa62318r/GDKUA001_lrecl80-FB.txt
/*
//LOCAL    DD   DISP=(NEW,CATLG),SPACE=(CYL,(5,5),RLSE),RECFM=FB,
//              LRECL=80,BLKSIZE=27920,VOL=SER=D9XWRK,
//              DSN=OA62318.CLDAU001.LRECL80.SAM.RECFMFB
****** **************************** Bottom of Data ***************************
```

Interactions & Dependencies

- Software Dependencies
  - IBM Cryptographic Service Facility (ICSF) CSFSERV and CSFKEYS classes
  - System SSL services – HTTPS communication
  - No optional features required.
- Hardware Dependencies
  - None
- Exploiters
  - None.

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level: No
- No toleration/coexistence PTFs
- The GDKUTIL

Installation & Configuration

- Included in z/OS 3.1 base
- APAR OA62318
    - Rolled down to z/OS 2.4 and z/OS 2.5 – No IPL required

- Configure the CSFKEYS general resource class to protect the keylabels for the encryption keys:
    a) The CSFKEYS general resource class must be active and RACLISTed.
    b) The ICSF segment of the CSFKEYS class profile CSF-PROTECTED-KEY-TOKEN (or its generic equivalent) must contain SYMCPACFWRAP(YES).
    c) The user's id must have READ access to the CSF-PROTECTED-KEY-TOKEN profile (or its generic equivalent).
    d) Define a profile for CSFKEYS resources beginning with GDK.** with a universal access (UACC) of NONE along with ICSF(SYMPACFWRAP(YES) SYMCPACFRET(YES)).
    e) The user's id must have READ access to the new CSFKEYS profile for resources beginning with GDK.<userid>.** along with ICSF(SYMPACFWRAP(YES) SYMCPACFRET(YES)).
    f) The security administrator or person who will be entering cloud provider keys must have UPDATE access to the new CSFKEYS profile for resources beginning with GDK.<userid>.**

Example z/OS Security Server RACF commands for the keylabel protection

```
 /* Define a generic label with UACC(NONE) so default access is NONE */
RDEFINE CSFKEYS GDK.** UACC(NONE) ICSF(SYMCPACFWRAP(YES) SYMCPACFRET(YES))

 /* Define a generic label specific to the CDAUSER               */
RDEFINE CSFKEYS GDK.CDAUSER.** UACC(NONE) ICSF(SYMCPACFWRAP(YES) SYMCPACFRET(YES))

 /* Permit the CDAUSER to their keylabels                        */
PERMIT GDK.CDAUSER.** CLASS(CSFKEYS) ID(CDAUSER) ACCESS(UPDATE)

SETROPTS RACLIST(CSFKEYS) CLASSACT(CSFKEYS) REFRESH
```

Installation & Configuration (cont)

- Ensure access to the required ICSF entry points. The user must have at least READ authority to the following CSFSERV Class resources:
  - CSFKGN
  - CSFRNGL
  - CSFKRD
  - CSFKRC2
  - CSFOWH

- CDA uses HTTPS connections with the remote Cloud Object Storage Server. The System SSL processing performed may require some setup. Details can be found in the z/OS Cryptographic Services System SSL Programming manual in the RACF CSFSERV resource requirements section. https://www.ibm.com/docs/en/zos/2.5.0?topic=ssl-racf-csfserv-resource-requirements

Installation & Configuration (cont)

- Configure CDA for system general use.
  a) Copy IBMCOS.json from /usr/lpp/dfsms/gdk/samples/providers/ to /usr/lpp/dfsms/gdk/providers/ (See note below in the Provider File subsection). Permissions should be set to 644
  b) Rename IBMCOS.json to any 20 character or less name, keeping the .json suffix.
  c) Modify the <provider>.json file to suit the Cloud Object Storage server you will use. This information should come from the administrator of the Cloud Object Storage server. (See the Provider File section below for more details on individual key/value pairs.)
     i) Change the "host" value to be the url for the Cloud Provider server.
     ii) Change the “port” value if necessary.
     iii) Change the “region” value if necessary.
     iv) Change the “sslCipher” value if the Cloud Provider server uses other SSL Ciphers.

Installation & Configuration (cont)

• Configure a RACF key ring for the Cloud Object store for z/OS as a client for the TLS/SSL traffic. You may create a virtual key ring. (See the “RACF and Digital Tokens” chapter in the z/OS Security Server RACF Administrator’s Guide for more details.)

1) Obtain the Root CA certificate of the target Cloud Object server. (One way is to use a browser, entering the Cloud Server url, and clicking on the lock icon to download the Root CA certificate to a local PC, followed by transferring the certificate to a data set on z/OS. *** Make sure you trust the Root CA ***)

2) Use RACDCERT ADD to add the Root CA certificate of the Cloud Provider server under CERTAUTH so that it is considered in the CERTAUTH’s virtual key ring.

3) Ensure the users of the CDA services have READ access to the virtual key ring where the certificates are stored. Only secure (HTTPS) connections are supported.

4) If the virtual key ring name is not *AUTH*/*, then update the provider file sslKey value to be the virtual key ring name used.

Example z/OS Security Server RACF commands to set up virtual key ring for the Cloud Object Store client.

```
 /* Define the following profile in the FACILITY class if it does
    not exist yet. */
RDEFINE FACILITY IRR.DIGTCERT.LISTRING  UACC(READ)

 /* Add the Root CA certificate of the remote Cloud Server under CERTAUTH.
    It is stored in the HLQ.ROOTCA.CLOUD data set.      */
RACDCERT     CERTAUTH     +
          ADD('HLQ.ROOTCA.CLOUD') +
          WITHLABEL('CLOUD') TRUST

 /* Refresh */
SETROPTS RACLIST (DIGTCERT) REFRESH

 /* Make sure the certificate was added correctly */
RACDCERT CERTAUTH LIST(LABEL(‘CLOUD'))
```

Installation & Configuration (cont)

- CDA Authorization Panels: 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

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

- User’s of CDA services require OMVS home directories

  a) The users RACF ID must have an OMVS segment defining the home directory.

User Configuration (cont)

<u>User Configuration Quick-Start</u>

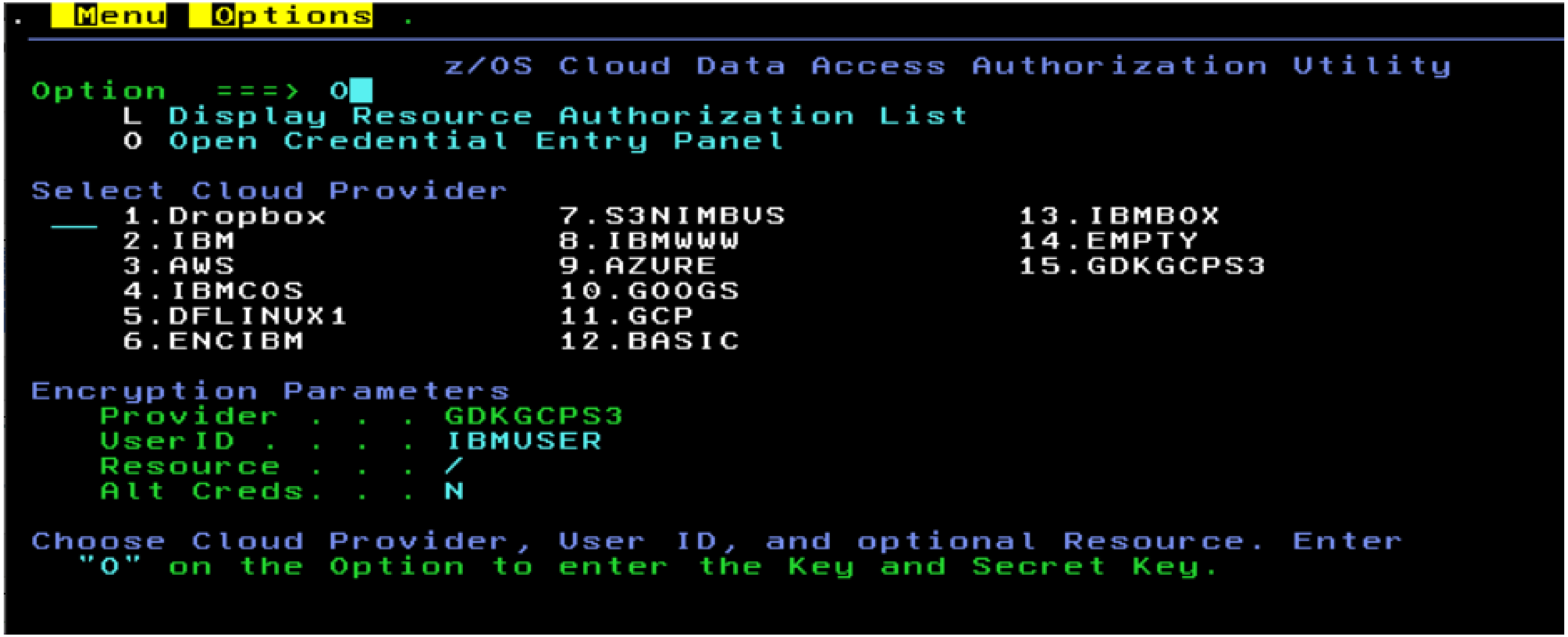
- Configure CDA for the user's environment.
  a) Create ~/gdk/ in the user’s UNIX home directory. Change the permissions to 700.
  b) Create ~/gdk/providers/ in user's UNIX home directory. Change the permissions to 700.
  c) Copy /usr/lpp/dfsms/gdk/samples/gdkkeyf.json to ~/gdk/gdkkeyf.json . Change the permissions to 600.
  d) (optional) The user may have a local config file in their ~/gdk/ directory. Copy /usr/lpp/dfsms/gdk/samples/gdkconfig.json to ~/gdk/config.json . Change the permissions to 600.
  e) (optional) The user may have a local provider file in their ~/gdk/providers/ directory if they want to use a modified provider definition. Copy /usr/lpp/dfsms/gdk/providers/*.json to ~/gdk/providers/ and change the permissions to 600.

- Use CDA Authorization Panel to store the Cloud Credentials associated with the user. (See the “Cloud Data Access Cloud Credential storage” section below for a detailed description and walkthrough.)

```
EX ‘SYS1.SAXREXEC(GDKAUTHP)’
```

  a) Enter the number of the Cloud Provider you are entering the credentials for
  b) Enter ‘O’ on the Option prompt to open the Credential Entry Panel
  c) Enter the Access Key (or userid) for the Cloud Object Storage server
  d) Enter the Secret Access Key (or password) for the Cloud Object Storage server
  e) Enter ‘S’ on the Option prompt to save the credentials

User Configuration Cloud Credentials



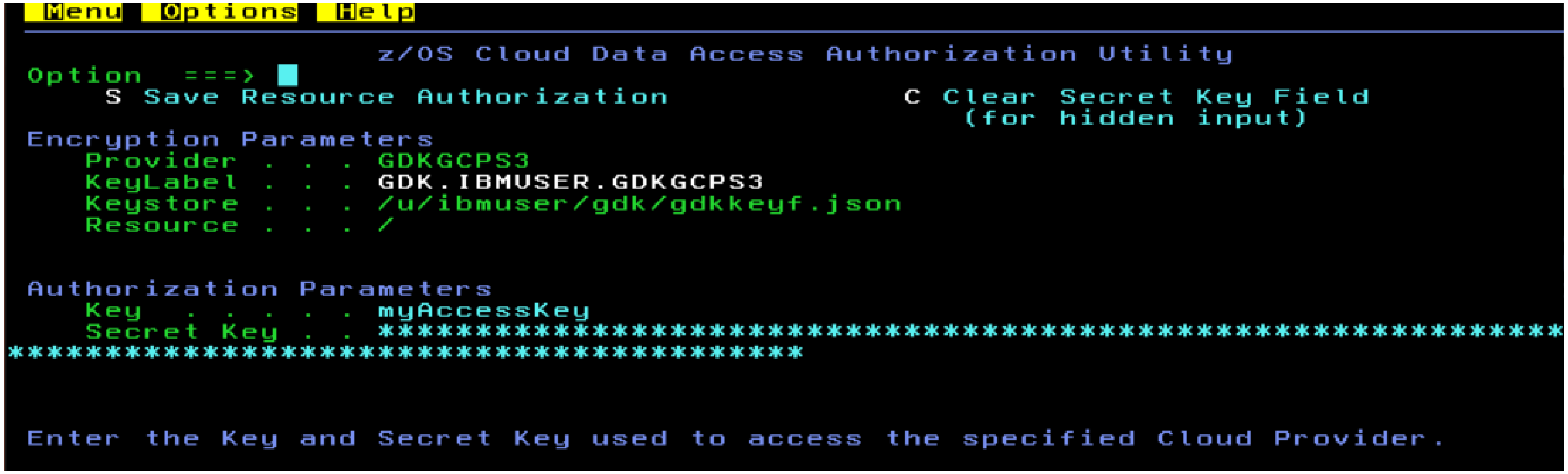
```
.  Menu  Options  .
                          z/OS Cloud Data Access Authorization Utility
Option  ===> O
    L Display Resource Authorization List
    O Open Credential Entry Panel

Select Cloud Provider
 __ 1.Dropbox                7.S3NIMBUS              13.IBMBOX
    2.IBM                    8.IBMWWW                14.EMPTY
    3.AWS                    9.AZURE                 15.GDKGCPS3
    4.IBMCOS                 10.GOOGS
    5.DFLINUX1               11.GCP
    6.ENCIBM                 12.BASIC

Encryption Parameters
   Provider . . .  GDKGCPS3
   UserID . . . .  IBMUSER
   Resource . . .  /
   Alt Creds. . .  N

Choose Cloud Provider, User ID, and optional Resource. Enter
  "O" on the Option to enter the Key and Secret Key.
```

User Configuration Cloud Credentials



```
 Menu  Options  Help
                  z/OS Cloud Data Access Authorization Utility
Option  ===>
    S Save Resource Authorization                 C Clear Secret Key Field
                                                    (for hidden input)
Encryption Parameters
   Provider . . . GDKGCPS3
   KeyLabel . . . GDK.IBMUSER.GDKGCPS3
   Keystore . . . /u/ibmuser/gdk/gdkkeyf.json
   Resource . . . /


Authorization Parameters
   Key  . . . . . myAccessKey
   Secret Key . . ****************************************************************
*****************************************



Enter the Key and Secret Key used to access the specified Cloud Provider.
```

Overview – ZRM-3340 Externalize CDA APIs

- Who (Audience)
  - z/OS Application Programmers using C, Assembler, or COBOL, that wish to interact with Cloud Objects
- What (Solution)
  - New I/O APIs are introduced, GDKGET, GDKWRITE, GDKLIST, GDKDEL
  - New credential management APIs are introduced, GDKKEYAD, GDKKEYSR, GDKKEYDL, GDKKEYGR
  - Supported Authentication mechanisms extended: Azure, OAUTH_2, BASIC, TEMPAUTH, NONE
- Wow (Benefit / Value, Need Addressed)
  - z/OS Application Programmers can code to a simple API to interact with Cloud Objects in a provider agnostic way. Different Cloud Object Storage providers can be accessed with the same API, enabling the application programmer to be relieved of worry about how to interact with that cloud provider.

Usage & Invocation - GDKGET

- Application programmers can retrieve a Cloud Object by calling the GDKGET API.
  - cloudProvider – Name of <provider>.json file in ~/gdk/providers/ directory
  - objectName – name of Object in the Cloud including the bucket name
  - dataLocationType – buffer, path, exit
  - dataLocation – pointer to the buffer, exit, or z/OS UNIX file or Data Set name
  - dataLocationLen – length of stuff that dataLocation points to.
  - optionalParmStructPtr – Structure that maps out key/value pairs that can modify CDA processing

```
gdkget (retCodeAddr,
        cloudProvider,
        objectName,
        dataLocationType,
        dataLocation,
        dataLocationLen,
        optionalParmStructPtr);
```

Usage & Invocation - GDKWRITE

- Application programmers can write to a Cloud Object by calling the GDKWRITE API.
  - cloudProvider – Name of <provider>.json file in ~/gdk/providers/ directory
  - objectName – name of Object in the Cloud including the bucket name
  - dataLocationType – buffer, path, exit
  - dataLocation – pointer to the buffer, exit, or z/OS UNIX file or Data Set name
  - dataLocationLen – length of stuff that dataLocation points to.
  - optionalParmStructPtr – Structure that maps out key/value pairs that can modify CDA processing

```
gdkwrite (retCodeAddr,
          cloudProvider,
          objectName,
          dataLocationType,
          dataLocation,
          dataLocationLen,
          optionalParmStructPtr);
```

Usage & Invocation - GDKDEL

- Application programmers can delete a Cloud Object by calling the GDKDEL API.
  - cloudProvider – Name of <provider>.json file in ~/gdk/providers/ directory
  - objectName – name of Object in the Cloud including the bucket name
  - optionalParmStructPtr – Structure that maps out key/value pairs that can modify CDA processing

```
gdkdel (retCodeAddr,
         cloudProvider,
         objectName,
         optionalParmStructPtr);
```

Usage & Invocation - GDKLIST

- Application programmers can request a list of Objects in a bucket by calling the GDKLIST API.
  - cloudProvider – Name of <provider>.json file in ~/gdk/providers/ directory
  - bucketName – name of Bucket in the Cloud.
  - buffer – pointer to a buffer of storage to be filled with the results
  - bufferLen – length of stuff that buffer points to.
  - optionalParmStructPtr – Structure that maps out key/value pairs that can modify CDA processing

```
gdklist (returnCodeAddr,
         cloudProvider,
         bucketName,
         buffer,
         bufferLen,
         optionalParmStructPtr);
```

Usage & Invocation - GDKMSGTR

- Application programmers can request a translation of a CDA return code into message text by calling the GDKMSGTR API.
  - CDAreturnCodeVal – integer return code value to be translated.
  - outputMessageBuff – pointer to a buffer of storage to be filled with the results
  - outputMessageBuffLen – length of stuff that buffer points to.
  - optionalParmStructPtr – Structure that maps out key/value pairs that can modify CDA processing

```
gdkmsgtr (returnCodeAddr,
          CDAreturnCodeVal,
          outputMessageBuff,
          outputMessageBuffLen,
          optionalParmStructPtr);
```

Usage & Invocation - GDKGETP

- Application programmers can request a list of Cloud Provider names that are available to use. (Combined list from user’s ~/gdk/providers/ and System default dir)
  - cloudProviderBufAddr – pointer to a buffer of storage to be filled with the results
  - cloudProviderBufLen – length of stuff that buffer points to.
  - optionalParmStructPtr – Structure that maps out key/value pairs that can modify CDA processing

```
gdkgetp (retCodeAddr,
         cloudProviderBufAddr,
         cloudProviderBufLen,
         optionalParmStructPtr);
```

Usage & Invocation - GDKKEYSR

- Application programmers can retrieve the saved Cloud Credentials. Current MVS user is used to find keyfile, and decrypt data using user’s ICSF Keylabel.

```
gdkkeysr (retCodeAddr,
          cloudProvider,
          resource,
          keyValue,
          keyValueBufLen,
          accessKeyValue,
          accessKeyValueBufLen,
          tenantValue,
          tenantValueBufLen,
          useridValue,
          useridValueBufLen,
          passwordValue,
          passwordValueBufLen,
          accessUrlValue,
          accessUrlValueBufLen,
          optionalParmStructPtr);
```

Usage & Invocation - GDKKEYAD

- Application programmers can store Cloud Credentials. Current MVS user is used to find keyfile, and encrypt credentials using a generated AES256 data key saved in ICSF under a Keylabel.
  - Access Key, Secret Access Key, Userid, Password, Tenant – Encrypted
  - Access URL – not encrypted

```
gdkkeyad (retCodeAddr,
          cloudProvider,
          resource,
          keyValue,
          accessKeyValue,
           tenantValue,
           useridValue,
           passwordValue,
           accessUrlValue,
          optionalParmStructPtr);
```

Usage & Invocation - GDKKEYDL

- Application programmers can delete Cloud Credentials from the user’s keyfile. ICSF Keylabel is also removed from CKDS. Current MVS user is used to find keyfile

```
gdkkeydl (retCodeAddr,
           cloudProvider,
           resource,
           optionalParmStructPtr);
```

Usage & Invocation - GDKKEYGR

- Application programmers can retrieve a list of the Cloud Credentials saved by resource from the user’s keyfile. (Doesn’t return actual credentials, just: Credentials for /. Credentials for /special/ .) Current MVS user is used to find keyfile

```
gdkkeydl (retCodeAddr,
           cloudProvider,
           resource,
           optionalParmStructPtr);
```

New Authentication Methods

- AZURE – Can authenticate with Microsoft Azure Cloud
- OAUTH_2 – Can authenticate with providers that use OAUTH_2.
  - Access Token
  - Client ID/Client Secret
  - JSON Web Token (JWT)
- BASIC – userid:password base64 encoded
- TEMPAUTH – OpenStack SWIFT to retrieve an Access Token
- NONE – Don’t use credentials – Good to retrieve Web objects

Documentation

- Full documentation found in z/OS V3R1 MVS Programming: Callable Services for High Level Languages.
  - Details on Branch Entry API
  - Language Environment requirements
  - Linking with SYS1.CSSLIB(GDKCSS/GDKCSSNL)
  - Return code meanings
  - Accepted OptionalParms
  - Provider file JSON schema

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - DFSMSdfp CDA Cloud Object Utility (GDKUTIL)

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level:
  - Yes. New authentication types not supported pre-z/OS 3.1
- List any toleration/coexistence APARs/PTFs.
  - None
- List anything that doesn't work the same anymore.
  - None

Installation & Configuration

- Additional provider files can be found in /usr/lpp/dfsms/gdk/samples/providers/
- C language sample programs can be found in SYS1.SAMPLIB(GDKCX1) and SYS1.SAMPLIB(GDKCX2)
- CDA ISPF Authorization Utility will display fields appropriate to the Authentication method.

Summary

- Batch JCL can easily incorporate Cloud Objects into data on z/OS by import/export.
- Application programmers can implement their own specific retrieval of data from Cloud Object storage in a simple way, irrespective of the target cloud object store.

Appendix

- z/OS MVS Programming: Callable Services for High Level Languages
- z/OS DFSMSdfp Utilities
- z/OS MVS System Messages, Vol 5 (EDG-GLZ)