z/OS DFSMSdfp CDA Cloud Object Utility

What follows are the planned updates to the z/OS publications for the Cloud Object Utility introduced in APAR OA62318.

Updated for OA63027 (10/31/2022)

contact: Andrew Wilt (anwilt@us.ibm.com)

1 Publication Changes:

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

z/OS MVS Programming: Callable Services for High Level Languages is updated to add a new chapter after Part 10: “SMF Services” as follows:

SA23-1377

1.1.1 Part 11: Cloud Data Access Services

The z/OS DFSMSdfp CDA component provides an environment that other z/OS products may utilize in communicating with Cloud Object Storage.

1.1.1.1 Introduction to z/OS DFSMSdfp Cloud Data Access

Cloud Object Storage provides a method to access data via RESTful APIs. Different Cloud Object Storage providers utilize different methods for authentication as well as have different specifics for the REST APIs for actions like READ or STORE an object. Additionally, a user’s Cloud Credentials for a specific Cloud Object Storage provider should be stored securely so that user does not have to enter the credentials for every communication with the cloud.

1.1.1.2 Cloud Data Access Configuration

Some configuration needs to be performed before using a program that utilizes the Cloud Data Access services. The RACF (or equivalent) userid associated with the environment using a program that uses CDA services must have an OMVS segment with a home directory defined. A gdk/ directory must be created in the user’s home directory. (Referred to as ~/gdk/) Permissions for the user’s ~/gdk/ directory should be set so that only the user has read and write authority to the files and directories within.

1.1.1.2.1 System Administrator Configuration Quick-Start

The Quick-Start guide below is intended to give a high-level view of the steps needed to set up for usage of Cloud Data Access services. Details of the steps are described in the sections below.

1) 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
```

2) 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

3) 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

4) 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.

5) 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.)
   a) 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 ***)
   b) 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.
   c) 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.
   d) 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'))
```

6) 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.

7) User's of CDA services require OMVS home directories
   a) The users RACF ID must have an OMVS segment defining the home directory.

1.1.1.2.2 User Configuration Quick-Start

1) 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.

2) 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

1.1.1.2.3 Cloud Data Access Files

DFSMSdfp CDA utilizes several files during its processing. They are:

- key file - Contains the encrypted Cloud Credentials for the user
- config file - Contains some configuration used during the saving of the Cloud Credentials
- provider file - Contains configuration specific to a specific Cloud Object Store provider

1.1.1.2.4 Key File

The cloud credentials to be used will be stored in a Key File named gdkkeyf.json in the ~/gdk/ directory. Before using the Cloud Data Access Authorization Panel described below in the **Cloud Data Access Cloud Credential storage** section, the /usr/lpp/dfsms/gdk/samples/gdkkeyf.json file should be copied to the user's ~/gdk/ directory.

If a Security Administrator is using the z/OS Cloud Data Access Authorization Utility to store the Cloud Credentials on the behalf of the user, then the Security Administrator should also have read/write permission to the ~/gdk/gdkkeyf.json file

1.1.1.2.5 Config File

The config file contains some configuration key:value pairs that the Cloud Data Access services may use during processing. Some programs that use the CDA services may request that the config file not be used during processing, however.

The System Administrator may copy the sample config file from /usr/lpp/dfsms/gdk/samples/gdkconfig.json to /usr/lpp/dfsms/gdk/config.json . If not explicitly told to not read the config file, the CDA services will first look in the user's directory for a config file (~/gdk/config.json), and if not found, it will look in the CDA global directory (/usr/lpp/dfsms/gdk/config.json) for a config file.

A user may wish to copy the /usr/lpp/dfsms/gdk/samples/gdkconfig.json file to their home directory ~/gdk/config.json so it can be modified for additional logging or debugging of processing.

Possible key:value pairs are:

- log-level - During CDA services processing different levels of logging are allowed.
    - ERROR - Only Error messages are logged (default)
    - WARNING - Warning messages and above are logged.
    - NOTICE - Notice messages and above are logged.
    - INFO - Informational messages and above are logged.
    - DEBUG - Flow and processing messages are logged.
    - NONE - No logging is performed by the CDA services.
- web-toolkit-logging - Request logging from the z/OS Client Web Enablement Toolkit
    - true - log messages
    - false - Don't log messages
- translation - Whether to request conversion of text data during READ or WRITE
    - true - Convert EBCDIC to ASCII on WRITE to Cloud or ASCII to EBCDIC on READ from Cloud
    - false - Don't convert data. (default)
- allow-no-CEX - If no Crypto Express card is available on the system, then the user accepts the reduced security of having the encryption keys stored in the clear in the ICSF CKDS
    - true
    - false (default)

1.1.1.2.6 Provider File

The specifics of how CDA services should interact with a Cloud Object Storage provider are detailed in a provider file. Each Cloud Object Storage provider that will be used on the system should be placed in a different provider file. Each provider file has a .json suffix, and is stored in either the users home directory (~/gdk/providers/), or in the CDA default location (/usr/lpp/dfsms/gdk/providers/). CDA services will first examine the user's ~/gdk/providers/ directory for a provider file ending in .json. If it no ~/gdk/providers/ directory is found, then /usr/lpp/dfsms/gdk/providers/ will be searched for the provider file. The maximum length of the provider name is 20 characters, not including the .json suffix. The provider names are case sensitive, so the name passed to the CDA program must match the name found in the gdk/providers directory.

The System Administrator may modify and place provider files for the Cloud Object Storage providers that can be used into the /usr/lpp/dfsms/gdk/providers/ directory. The permissions of these files should be read only for end users (rw-r--r-- or chmod 644).

Note:

The files in the /usr/lpp/dfsms/gdk/providers directory are intended to be default versions tailored specifically to your Cloud Object Storage provider. Users may copy the tailored version to their personal directory if they want to make modifications.

Many sites copy the ZFS that contains /usr/lpp/dfsms/gdk/providers across to each system in the sysplex, thus losing any tailored files placed in /usr/lpp/dfsms/gdk/providers. To avoid this, you may create a ZFS data set and mount it to the /usr/lpp/dfsms/gdk/providers directory, placing the specifics in the SYS1.PARMLIB(BPXPRMxx) member so it is automatically mounted during IPL.

Sample provider files may be found in /usr/lpp/dfsms/gdk/samples/providers/ .

- IBMCOS.json describes an IBM Cloud Object Storage provider
- S3CLOUD.json describes an AWS S3 Cloud Object Storage provider

A provider file is a JSON object. The provider file should be modified for the particular Cloud Object Storage provider available to the z/OS LPAR. Usually only the host and region need to be modified, and possibly others depending on the needs of the environment.

- name - Descriptive name for this cloud provider - Not used by CDA services
- host - URI for the Cloud Object Storage provider
- region - region identifier such as us-west-1
- port - HTTPS port number used if not default 443
- httpMechanism - Must be set to "HTTPS"
- sendTimeout - Length of time in seconds for send request to timeout (not required)
- receiveTimeout - Length of time in seconds for a receive to timeout (not required)
- IPStack - Name of alternate z/OS IP Stack to use in HTTP communication (not required)
- sslVersion - Type of SSL to use
    - TLSV10
    - TLSV11
    - TLSV12
- sslCiphers - Override the SSL Ciphers to be used in the communication (not required unless the https connection uses others than the default) SSL Cipher Suite details can

be found in the “Cipher suite definitions” chapter of the z/OS Cryptographic Services System SSL Programming publication.

- sslKey - Override the default virtual key ring name of *AUTH*/*
- encode - what type of url-encoding to use (not required)
  - special - Use AWS url-encoding for object names
- encodeUrlChars - Characters in the object name that should be url-encoded by CDA.
- errorUrlChars - Characters in the object name that should result in a failure if specified. Note that the backslash character is a special JSON character, and must be escaped by a backslash character if you want to specify it.
- authentication -
  - model : AWS4 - Use AWS4 authentication model when communicating with the Cloud Object Storage provider
- supportedOperations - Array of operations objects. “name” is one of { GETOBJECT, GETLARGEOBJECT, WRITEOBJECT, WRITELARGEOBJECT, LISTOBJECT, DELETEOBJECT }
  - name - GETOBJECT
  - apiEndpoint - how to build url for request
    - <HOST><GDK_OBJECT_NAME>
  - httpMethod - { GET, PUT, POST, DELETE, HEAD }
  - multipartChunksize - The size of each chunk in bytes to request when retrieving an object in a GETLARGEOBJECT operation.
  - actions - An array of operation objects that describe how to perform a multi-step request such as GETLARGEOBJECT, or WRITELARGEOBJECT
  - requestParameters - JSON object describing parameters on the request
    - mechanism - details about how to build parts of request
      - HEADER - specifics about what headers need to be included. (Can be multiple)
      - MESSAGE_BODY - Request body content
    - descriptor - how to build the mechanism
      - “ some data “ passed as-is unless <text> is found such as DATE_ISO_8601
      - <GDK_DATA> - The data passed into CDA services
    - contentType - how to set content type for HTTP request
      - text/plain
      - application/xml
      - application/octet-stream
  - responseResults - allows definition of how to parse the response body from a LISTOBJECT request.

A GETLARGEOBJECT operation is optional, but extends the functionality described by a GETOBJECT operation. It is expected to be made up of a name key:value pair, and an actions array. The actions array must have the following objects:

- name: getSize - Description on how to retrieve the size of a single Cloud Object
- name: data - Description how to retrieve the data
- The data action may have an optional multipartChunksize: “number” that overrides the default 8MB size for each retrieve that is used to retrieve the entire Cloud Object

A WRITELARGEOBJECT operation is optional, but extends the functionality described by a WRITEOBJECT operation. It is expected to be made up of a name: WRITELARGEOBJECT pair and an actions array. The actions array consists of the following objects.

- name: “init” - (optional) Describes the action to take to initialize a Multipart Upload sequence.
- name: “data” - (required) Describes the action to upload a part of the data to be sent. The chunk size for each part is 8MB, except for the last.
- name: “complete” - (optional) Describes the action to complete a Multipart Upload sequence
- name: “error” - (optional) Describes the action needed to Abort a Multipart upload so that individual parts do not take up space when an error occurred, and the Multipart upload was unable to complete successfully.

1.1.1.3 Cloud Data Access Cloud Credential storage

For the CDA services to perform its communication with the Cloud Object Storage server, it needs to use the Cloud Credentials of the user. Additionally, programs using the CDA services do not want to ask the user to enter the Cloud Credentials every time they use the CDA services. Therefore, the Cloud Credentials need to be available to the CDA services in a secure format.

The cloud credentials that will be used by the program when executed by the user are entered using the z/OS Cloud Data Access Authorization Utility. The Cloud Credentials are encrypted using a random key that is stored in the IBM Cryptographic Service Facility (ICSF) CKDS. The encrypted Cloud Credentials are written to the user’s gdkkeyf.json file in the ~/gdk/ directory.

The encryption key is stored in ICSF under a keylabel of the form, GDK.<userid>.<provider>.Annnn . Where <userid> is the SAF userid for the user, and <provider> is the Cloud Provider the Cloud Credentials are associated with. A suffix, Annnnn where nnnnn is a 5 digit number used internally by CDA, is appended. The keylabel is protected by the CSFKEYS general resource class, which must have SYMCPACFWRAP(YES) in the ICSF segment. The userid must have READ access to the keylabel profile. GDK.<userid>.* is recommended.

If a Security Administrator is using the z/OS Cloud Data Access Authorization Utility to store the Cloud Credentials on the behalf of the user, then the Security Administrator should also have read/write permission to the ~/gdk/gdkkeyf.json file.

Note:
To ensure the highest level of security of the encryption key, the system should have a Crypto Express card installed so that the encryption key stored in the ICSF CKDS is wrapped by the Crypto Express Card’s master key. If no card is available, the config sample file may be copied from /usr/lpp/dfsms/gdk/samples/gdkconfig.json and placed in the user’s home directory as

~/gdk/config.json . The key:value pair of “allow-no-CEX”: true can be added to the config.json file to indicate that the user accepts the reduced security of the encrypted Cloud Credentials when no Crypto Express card is available for use. When no Crypto Express card is available, and the “allow-no-CEX”:true key:value pair exists, this indicates the user accepts the data encryption key for the encrypted cloud credentials being stored in the clear in the ICSF Cryptographic Key Data Set (CKDS).

Ensure that the security administrator or user who will be entering the cloud provider keys has sufficient authority to write to the user’s gdkkeyf.json file (~/gdk/gdkkeyf.json) and UPDATE permission to the CSFKEYS profile for resources beginning with GDK.<userid>.* .

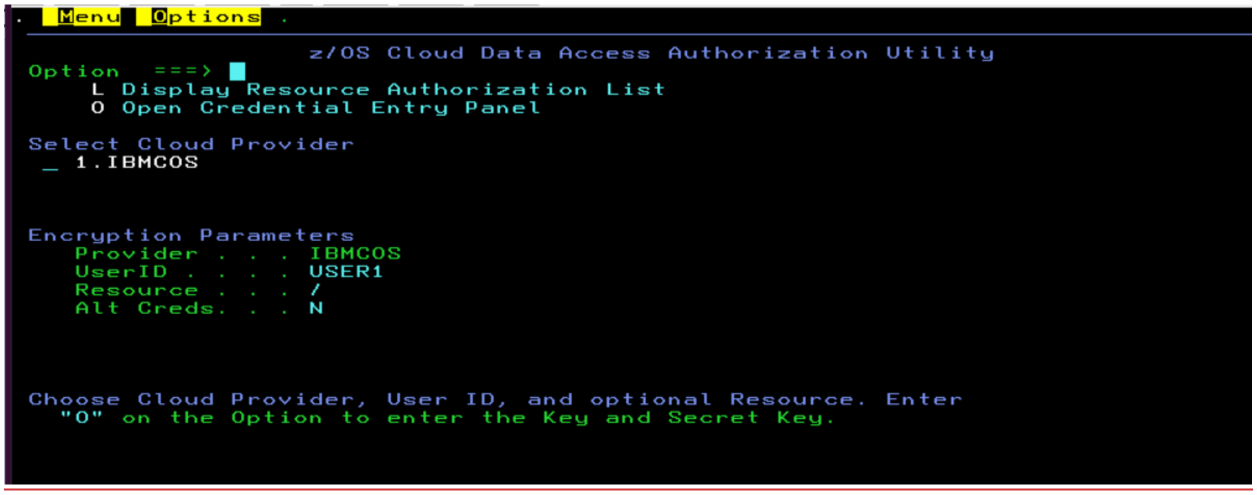
The process of saving the cloud credentials entails encrypting the cloud credentials using ICSF services. Refer to the z/OS Cryptographic Services ICSF Administrator’s Guide for more information on CCA and ICSF entry points. The user must have at least READ authority to the following CSFSERV Class resources:

- CSFKGN
- CSFRNGL
- CSFKRD
- CSFKRC2
- CSFOWH

The CDA Authorization Utility may be invoked from the ISPF Command Shell using the following command:

```
EX 'SYS1.SAXREXEC(GDKAUTHP)'
```

This will start a CDA panel where the Cloud Credentials will be encrypted and saved.



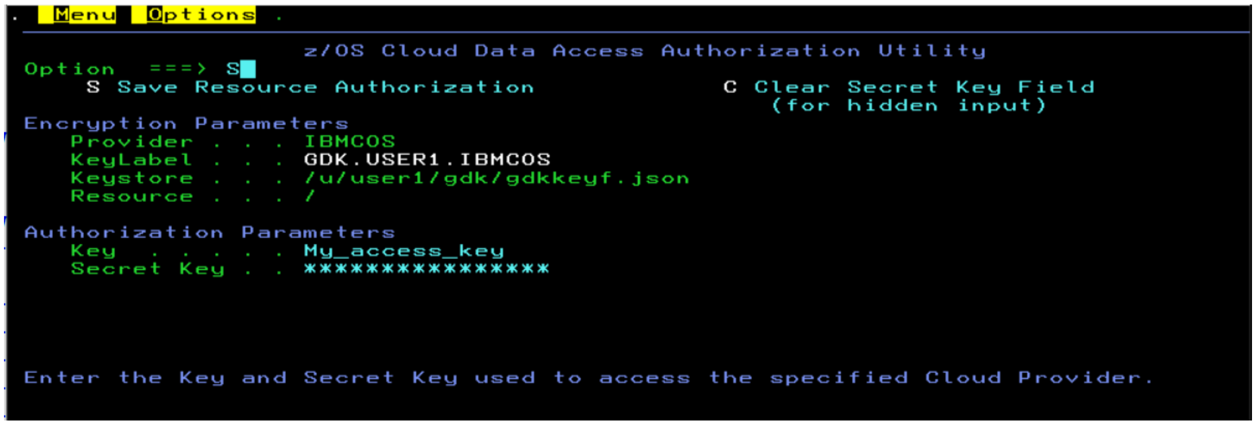
```
 .  Menu  Options  .
                    z/OS Cloud Data Access Authorization Utility
 Option   ===>
        L Display Resource Authorization List
        O Open Credential Entry Panel

 Select Cloud Provider
  _  1.IBMCOS


 Encryption Parameters
    Provider . . .  IBMCOS
    UserID . . . .  USER1
    Resource . . .  /
    Alt Creds. . .  N



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

1. The Cloud Provider names (ending in .json) are read from the user’s ~/gdk/providers/ directory and displayed (without .json). (If there is no ~/gdk/providers/ directory, then the default /usr/lpp/dfsms/gdk/providers/ directory will be used.)

2. Enter the RACF (or equivalent) user id that will be using the CDA Cloud Object Utility into the **UserID** field under the “Encryption Parameters” section.
3. Select the cloud provider associated with the key pair being added by entering the associated number under the “Select Cloud Provider” heading. The currently chosen provider will be displayed in the **Provider** field under the “Encryption Parameters” section.
4. If this key pair is intended to be used with a specific bucket, enter a ‘/’ followed by the bucket name in the **Resource** field under the “Encryption Parameters” section. Otherwise, simply enter a ‘/’ to indicate that this key pair is valid for any bucket associated with this cloud provider.

   Keys for accessing objects in buckets generally for the account are associated with the ‘/’ resource. Keys for accessing objects in specific buckets that have different can be added and associated with that /bucket_name. CDA services will attempt to utilize specific keys tied to buckets before utilizing the generic key for the provider.

   Note: Only 1 generic key is used per provider. If a second is entered, it will overwrite the first.
5. If this key pair is intended to be used as the Alternate Credentials by a user of the CDA APIs, enter ‘Y’ for the Alt Creds field.
6. Enter an ‘O’ on the top Option line to continue to the next panel


```
  Menu  Options  .
                          z/OS Cloud Data Access Authorization Utility
 Option  ===> S
     S Save Resource Authorization                        C Clear Secret Key Field
                                                             (for hidden input)
 Encryption Parameters
    Provider . . . IBMCOS
    KeyLabel . . . GDK.USER1.IBMCOS
    Keystore . . . /u/user1/gdk/gdkkeyf.json
    Resource . . . /

 Authorization Parameters
    Key . . . . . My_access_key
    Secret Key . . ****************




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

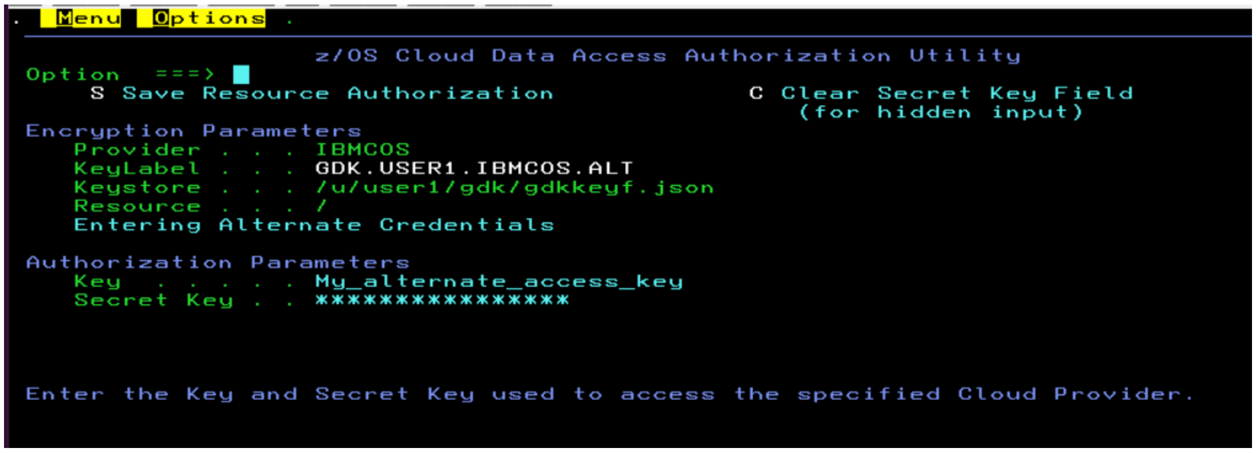
7. Enter the Key and Secret key values into the associated fields under the “Authorization Parameters” section. Press Enter. The characters are not echoed to the screen and are displayed as * when enter is hit.
8. Enter an ‘S’ on the top Option line to encrypt and save the key pair.

Note:

The first time this panel is executed, the user may receive the following warning messages:
`ERROR: getpwnam() error: EDC5121I Invalid argument.`
`ERROR: getpwnam() error: EDC5129I No such file or directory.`

This behavior is expected because the UserID field has not yet been populated. Once the user id being used has been specified here at least one time, the warning messages will no longer be displayed.

When entering Alternate Credentials, the KeyLabel is additionally appended with .ALT, and an informational line of “Entering Alternate Credentials” is displayed.



```
 .  Menu  Options  .
                    z/OS Cloud Data Access Authorization Utility
 Option  ===> 
     S Save Resource Authorization                C Clear Secret Key Field
                                                     (for hidden input)
 Encryption Parameters
    Provider . . .  IBMCOS
    KeyLabel . . .  GDK.USER1.IBMCOS.ALT
    Keystore . . .  /u/user1/gdk/gdkkeyf.json
    Resource . . .  /
    Entering Alternate Credentials

 Authorization Parameters
    Key  . . . . .  My_alternate_access_key
    Secret Key . .  ****************


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

1.1.1.3.1 Error Conditions

If an error occurs during the saving of the Cloud Credentials, an error message will be written indicating that it was unable to store the cloud credentials indicating the IBM Cryptographic Service Facility (ICSF) API that encountered the error along with the return code and reason code returned by the service. The return code and reason codes are documented in the ICSF and cryptographic coprocessor return and reason codes chapter of the z/OS Cryptographic Services ICSF Application Programmer’s Guide.

ICSF reason code	Description
0BFB	Received when the ICSF segment of the CSFKEYS class does not have SYMCPACFWRAP(YES)
3E80	Permission to the required CSFSERV Class is not READ or better.
3E84	Received when the RACF userid does not have permission to the necessary keylabel in the CSFKEYS class needed to store the encryption key for the encrypted Cloud Credentials
271C	Received when the ICSF Keylabel can’t be found. Usually the keylabel was mistakenly deleted from the ICSF CKDS. The Cloud Credentials must be saved again.

1.2 z/OS DFSMSdfp Utilities

z/OS DFSMSdfp Utilities is updated to add a new chapter, titled “GDKUTIL (Cloud Object Utility) Program” as follows:

SC23-6864

GDKUTIL (Cloud Object Utility) Program

GDKUTIL is a utility that is used to copy objects in Cloud Object Storage to z/OS. It can also be used to copy z/OS data sets or UNIX files to Cloud Object Storage.

Cloud Object Storage providers that support the Simple Storage Service (S3) protocol using AWS4 authentication can be targeted by the GDKUTIL program. The utility allows upload and download of data, with optional EBCDIC to UTF-8 conversion on the upload and UTF-8 to EBCDIC conversion on the download.

Environment requirements:

See the **Cloud Data Access Configuration** section in the z/OS MVS Programming: Callable Services for High Level languages for details regarding the environment required for the RACF user invoking the GDKUTIL program.

Saving the Cloud Credentials to be Used

Refer to the **Cloud Data Access Cloud Credential storage** section in the z/OS MVS Programming: Callable Services for High Level Languages for details regarding storing the Cloud Credentials to be used when communicating with the Cloud Object Storage.

Control

GDKUTIL is controlled by job control and utility control statements. The job control statements define the local z/OS UNIX file or data set, as well as the Cloud Object name to be used in the processing. The utility control statements are used to control the functions of the program.

Job Control

Table xx on page yy shows the job control statements for GDKUTIL.

Table xx

Statement	Use
JOB	Starts the job
EXEC	Specifies the program name (PGM=GDKUTIL), or if the job control statements reside in a procedure library, the procedure name.

SYSPRINT	Defines a sequential data set or UNIX file for Utility output messages. The data set or UNIX file can be written on a system output device, a tape volume, or a DASD volume
SYSIN	Defines the control data set, which contains the utility control statements. The data set normally resides in the input stream; however, it can be defined as a sequential data set, a member of a partitioned data set or PDSE, or a UNIX file. Any text in the last 8 characters of each record is ignored in case sequence numbers are used.
SYSOUT	Defines a sequential data set or UNIX file for DFSMSdfp CDA logging and error messages. The data set or UNIX file can be written on a system output device, a tape volume, or a DASD volume.
OBJNAME	Defines a sequential data set or UNIX file that contains the Cloud Object name to be used in the operation. It can be in-stream data (DD *), or point to a sequential data set or UNIX file. (Required)
LOCAL	Defines a data set or UNIX file that will be used in the operation. (Optional)
LOCNAME	Defines a sequential data set or UNIX file that contains the name of a z/OS data set or UNIX file to be used in the operation. It can be in-stream data (DD *, or point to a sequential data set or UNIX file. (Required if LOCAL does not exist.)

EXEC Statement

The EXEC statement specifies PGM=GDKUTIL. Any PARM parameter is ignored.

SYSOUT Statement

If the SYSOUT statement is omitted, one will be dynamically created if logging is requested.

SYSPRINT Statement

If the SYSPRINT statement is omitted, one will be dynamically created for any messages written.

OBJNAME Statement

The OBJNAME statement describes the location that holds the object name in Cloud Storage that will be used in the utility operation. The first non-blank record in a data set, member, or in-stream data, or non-blank line in a UNIX file is read and the content is used as the object name. The object name must start with a forward slash, followed by the bucket or container name. The bucket or container must already exist. The rest of the object name follows, starting with a forward slash. Additional forward slashes may be used to create a pseudo-directory structure. Refer to the cloud provider documentation for additional details. Leading blanks and trailing blank characters in the name are trimmed.

Most characters are acceptable. However, the following characters are unacceptable and may cause the request to fail:

- Backslash ("\")
- Left curly brace ("{")
- Non-printable ASCII characters (128–255 decimal characters)
- Caret ("^")
- Right curly brace ("}")
- Percent character ("%")
- Grave accent / back tick ("`")
- Right square bracket ("]")
- Quotation marks
- 'Greater Than' symbol (">")
- Left square bracket ("[")
- Tilde ("~")
- 'Less Than' symbol ("<")
- 'Pound' character ("#")
- Vertical bar / pipe ("|")

LOCAL Statement

The LOCAL statement describes a sequential z/OS data set or member of a Partitioned Data Set or PDS/E, or a z/OS UNIX file. For an UPLOAD command, the data in the z/OS UNIX file or data set will be read and placed into the cloud object. The z/OS data set or member must have a record format (RECFM) of F, FB, V, or VB, and logical record length greater than zero (LRECL). A RECFM=U data set may also be used. For a DOWNLOAD command, the data will be appended if DISP=MOD, or PATHOPTS=(OAPPEND) is used. If DISP=OLD, or PATHOPTS=(OTRUNC), the existing data will be overwritten.

Note: While a PDSE or PDS data set name may be specified for an UPLOAD command, this only opens the directory for read, and none of the members are processed.

Note: When specifying a RECFM=U data set for an UPLOAD, the CONVERT keyword should not be specified. When the CONVERT keyword is not specified, the raw data from each block is sent, and any record or block boundaries are lost. Likewise, during a DOWNLOAD command targeting a RECFM=U data set, the byte stream is not examined to guess at the record boundaries.

LOCNAME Statement

The LOCNAME statement describes the location that holds the z/OS sequential data set, PDS or PDS/E member, or version of a Generation Data Group (GDG) that will be used in the utility operation. The first non-blank record in a data set, member, or in-stream data, or non-blank line in a UNIX file is read and the content is used as the local name.

If the name does not start with a forward-slash, it is assumed to be a fully qualified Data Set name. UNIX file names must be an absolute path.

Additionally, the data set name notation for fopen() can be used. e.g.

- `//’FULLY.QUALIFD.NAME’`

- `//'PDSE.DATASET.NAME(MEMBER1)'`
- `//'GENER.ATION.DATASET(+1)'`

Details can be found in the z/OS XL C/C++ Programming Guide, Input and Output chapter, Performing OS I/O operations-> Opening Files -> Using fopen() or freopen().

https://www.ibm.com/docs/en/zos/2.4.0?topic=SSLTBW_2.4.0/com.ibm.zos.v2r4.cbcpx01/cbc1p210.htm

Note: If using a generation data set relative name and you want to create a new (+1) generation, this must be done by a previous step in the JCL JOB. The utility will not create a new generation from the name in the LOCNAME statement.

Note: Data Set attribute considerations
When specifying a z/OS Data Set for LOCAL or LOCNAME for a DOWNLOAD command without the CONVERT keyword, the Record Format should not be Variable. (RECFM=V or RECFM=VB) This is because in binary mode, there are no indicators in the data where a record ends, so the data is placed up to the maximum logical record length. If a Data Set with variable records was uploaded to a Cloud Object in binary mode, the indicators of where a record begins and ends are lost. Therefore, a download operation of that same data to a Variable record data set cannot tell when to start a new record in the data set. During a DOWNLOAD command, if the data set name specified on the LOCNAME DD does not exist, a data set will be created automatically using the defaults described in the fopen() defaults section of the “Input and Output” chapter of the z/OS XL C/C++ Programming Guide. These defaults may not meet your expectations for containing the data.

SYSIN Statement

GDKUTIL is controlled by the utility control statements shown in the table below.

Statement	**Use**
Commands	
UPLOAD	Requests the data in LOCAL or the z/OS UNIX file / Data Set named in the LOCNAME DD be sent to the Cloud Object Storage provider referenced on the PROVIDER keyword. The data will be stored in the object named by the OBJNAME DD. If the Cloud already has an object with that name, it will be overwritten without warning. The bucket must already exist. You may request the data be processed differently by specifying the CONVERT keyword. Minimum length UP
DOWNLOAD	Requests the data in the object named by the OBJNAME DD for the provider referenced on the PROVIDER keyword be written to the z/OS

	UNIX file, or Data Set/member referenced on the LOCAL or LOCNAME DD. If the LOCAL DD is used, and DISP=OLD is used, the data will be overwritten. If DISP=MOD is used, the data will be appended to the end. The request will fail if DISP=SHR is used. If LOCNAME is used and the z/OS UNIX file or data set exists, its data will be overwritten. If the z/OS UNIX file or data set does not exist, it will be created using the defaults described in the fopen() defaults section of the “Input and Output” chapter of the z/OS XL C/C++ Programming Guide. These defaults may not meet your expectations for containing the data. Minimum length DOWN. Note: If not transferring in text mode, and the target data set is RECFM=V or RECFM=VB, all records will be the same length, except possibly the last.
DELETE	Requests the object named by the OBJNAME DD for the provider referenced on the PROVIDER keyword be removed from the Cloud Object Storage bucket.
LIST	Requests a list of objects in the bucket named by the OBJNAME DD for the provider referenced on the PROVIDER keyword be written to the SYSPRINT DD.
Keywords	
PROVIDER(name)	Required. Indicates the name of a ~/gdk/providers/<provider_name>.json file that describes the Cloud Object Storage provider that will be used. The specified name is case sensitive and must match the case of the file in the providers directory. See the Provider File section of the Cloud Data Access Configuration chapter in the z/OS MVS Programming: Callable Services for High Level Languages manual for more details about the provider file. Alias: PROV(<provider_name>)
CONVERT	Optional. Indicates that the data should be converted from EBCDIC to UTF-8 on UPLOAD, or from UTF-8 to EBCDIC on DOWNLOAD. The data is read or written to the z/OS data set or UNIX file in text mode. In text mode during UPLOAD, when reading from a z/OS data set, each record has a newline character appended at the end of the record data. In text mode during DOWNLOAD, the newline characters are used as record delimiters, and are stripped before writing that record to the output data set. If not specified, the data is read/written in binary mode. No bytes are added or removed from the data during I/O.

	Minimum text CONV.
DELSRC	Optional. Indicates that the source should be deleted after a successful processing. If specified on an UPLOAD command, the utility attempts to remove the z/OS UNIX file or data set. If specified on a DOWNLOAD command, a request to delete the cloud object is sent. Only valid for UPLOAD and DOWNLOAD commands.
LOG(level)	Optional. Indicates an override of the logging level. The default level is ERROR. <level> may be one of: • NONE - No logging done, including errors. • ERROR (default) - Only Errors are logged • WARNING - Only Warnings and above are logged. • NOTICE - Only Notification types and above are logged. • INFO - Only Information types and above are logged. • DEBUG - All information is logged.
WEBTOOLKITLOG	Optional. Request logging from the z/OS Client Web Enablement Toolkit. This output is written to the SYSPRINT DD. This logging is useful in debugging errors occurring during communication with the Cloud Object Storage server. See the z/OS MVS Programming: Callable Services for High-Level Languages, z/OS client web enablement toolkit chapter for information about the HWTH_OPT_VERBOSE option. Abbreviation: WTKLOG
WEBTOOLKITLOGU	Optional. Requests the Unredacted logging output from the z/OS Client Web Enablement Toolkit. Abbreviation: WTKLOGU

GDKUTIL Examples

Example 1: Retrieve an object into a z/OS UNIX file

In this example a binary Cloud object is retrieved and stored into a z/OS UNIX file.

```
//DOWNCLD  EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN   DD *
 DOWNLOAD PROVIDER(IBMCOS)
/*
//OBJNAME  DD *
    /cloudbucket01/file1.jar
/*
//LOCNAME  DD *
/prod/input-files/cloud/file1.jar
/*
```

The file1.jar object in the cloudbucket01 bucket is specified as the object name on the instream DD for OBJNAME. The local name specified using the LOCNAME instream DD is in the local UNIX directory /prod/input-file/cloud/ with the filename being file1.jar. The SYSIN directs the action to be a Download of the object to the Local name from the Cloud provider specified in the IBMCOS.json file in the gdk/providers/ directory.

Example 2: Send a z/OS Sequential data set to a Cloud Object

In this example, a sequential data set with EBCDIC text data is sent to a cloud object with conversion to UTF-8. Note that the /cloudbucket01/ bucket must have been created previously.

```
//UPCLOUD  EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN   DD *
 UPLOAD PROVIDER(IBMCOS) CONVERT
/*
//OBJNAME  DD *
    /cloudbucket01/processing-report-20211018.txt
/*
//LOCAL  DD DISP=SHR,DSN=REPORTS.CICSPROC.D211018
```

1.3 z/OS MVS System Messages, Vol 5 (EDG-GLZ)

SA38-0672

z/OS MVS System Messages, Vol 5 (EDG-GLZ) is updated to add a new chapter, titled “GDK messages” as follows:

This topic contains the messages issued by DFSMSdfp CDA. The format of DFSMSdfp CDA messages is:

- GDKcnnnnx

where:

- GDK is the DFSMSdfp CDA identifier

- c is an alphanumeric identifier assigned to a component of DFSMSdfp CDA. The message identifiers are assigned as follows:

c value	Component
U	Cloud Object Utility

- nnnn is a 4 digit message number

- x indicates the severity of the error. The value of x can be:

x value	Explanation
I	Informational: No action is required.
E	Error: An error occurred. Some change must be made for a successful process.

1.3.1 GDKU0010E

GDKU0010E UNABLE TO OPEN SYSIN DD

Explanation

During utility initialization the SYSIN DD statement could not be opened to read the SYSIN commands and keywords.

System Action

The Task is not performed. The return code is 8.

Operator Response

None.

Programmer Response

Examine the JOB output for the required SYSIN DD statement.

Source

DFSMSdfp CDA

1.3.2 GDKU0011E

GDKU0011E {SYSIN|OBJNAME|LOCNAME} LRECL TOO SMALL <*size*>

Explanation

The Logical Record Length (LRECL) for the named DD statement is less than the minimum 72 bytes.

System Action

The Task is not performed. The return code is 8.

Operator Response

None.

Programmer Response

Ensure that the LRECL for the named DD statement is 72 bytes or more.

Source

DFSMSdfp CDA

1.3.3 GDKU0013I

GDKU0013I <DD name>: <contents of DD>

Explanation

The name of the Cloud Object or Local z/OS UNIX file or data set name is displayed.

DD name	may be OBJNAME, or LOCNAME

System Action

Processing continues.

Operator Response

None.

Programmer Response

None.

Source

DFSMSdfp CDA

1.3.4 GDKU0014E

GDKU0014E MISSING DD FOR <*command*>. <*missing_DD*> DD REQUIRED

Explanation

Required DD names for the requested command are missing for the requested task.

System Action

The Task is not performed. The return code is 8.

Operator Response

None.

Programmer Response

Ensure the *missing_DD* names are specified and contain the expected names or content.

Source

DFSMSdfp CDA

1.3.5 GDKU0015E

GDKU0015E PARSE ERROR: <*error_text*>

Explanation

Parsing of the SYSIN content found missing information required for the Task.

System Action

The Task is not performed. The return code is 8.

Operator Response

None.

Programmer Response

Ensure the SYSIN contents meet the required keyword specifications for the Task.

Source

DFSMSdfp CDA

1.3.6 GDKU0016E

GDKU0016E MISSING KEYWORD: PROVIDER(name)

Explanation

The request requires a Cloud provider name to be specified via the PROVIDER keyword.

System Action

The Task is not performed. The return code is 8.

Operator Response

None.

Programmer Response

Source

DFSMSdfp CDA

1.3.7 GDKU0100I

GDKU0100I {UPLOAD|DOWNLOAD|DELETE|LIST} PROCESSING SUCCESSFUL

Explanation

The requested command has completed successfully.

System Action

None.

Operator Response

None.

Programmer Response

Source

DFSMSdfp CDA

1.3.8 GDKU0101E

GDKU0101E ERROR DURING <*command*> REQUEST. GDKRC=<rc>: <description>

Explanation

During processing of the requested command, an error occurred.

System Action

The Task did not complete successfully. The return code is 8.

Operator Response

None.

Programmer Response

Using the GDKRC value and description of the error, correct the error and retry the command.

Source

DFSMSdfp CDA

1.3.9 GDKU0102E

GDKU0102E ERROR DURING DELSRC PROCESSING. {errno|GDKRC}=<val>: <description>

Explanation

The Task did not complete successfully. The return code is 8.

System Action

Operator Response

None.

Programmer Response

Source

DFSMSdfp CDA

1.3.10 GDKU0103I

GDKU0103I LIST OF BUCKET <bucketname> IN CLOUD <cloud>

Explanation

A LIST command was requested for the specified <bucketname> from the Cloud provider <cloud>. The following lines consist of a comma-separated list of object names along with the sizes.

System Action

None

Operator Response

None.

Programmer Response

None.

Source

DFSMSdfp CDA

1.3.11 GDKU0104I

GDKU0104I NO MATCHING OBJECTS FOR BUCKET <bucketname> IN CLOUD <cloud>

Explanation

A LIST command was requested for the specified <bucketname> from the Cloud provider <cloud>. The Cloud Object Storage server returned data indicating that nothing matched the request.

System Action

None

Operator Response

None.

Programmer Response

None.

Source

DFSMSdfp CDA

1.3.12 GDKU0105I

GDKU0105I DATA RETURNED FROM CLOUD <cloud>

Explanation

A LIST command was requested and the provider file requested the data returned be in text/plain format. Specifically, the LISTOBJECT action specified responseResults with a contentType of text/plain. The following lines consists of the raw information returned from the Cloud Object server for this request.

System Action

None

Operator Response

None.

Programmer Response

None.

Source

DFSMSdfp CDA