OA65990 Publication Updates

DFSMSdfp CDA

12/03/2024

Andrew Wilt

This document describes the updates to the z/OS 3.1 publications as a result of the new function delivered via OA65990.

1 Overview

1.1 User Actions

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

1.1.1 Provider File Updates

If you want to set a new local code page default, or remote code page default, an update to your provider file needs to be made at the top-level. The following new keys are recognized:

"localCodePage": "IBM-1047",

"remoteCodePage": "ISO8859-1",

For the value, specify the code page name or CCSID for the desired local code page default or the remote code page default.

Provider file excerpt

```
 "name": "IBMCOS",
 "host": "s3.us.cloud-object-storage.appdomain.cloud",
 "port": "443",
 "region": "us-standard",
 "httpMechanism": "HTTPS",
 "sslVersion": "TLSV12",
 "sslCiphers": "C013C014C027C028C02FC030",
 "encodeUrlChars": " &$@=;:+,?",
 "errorUrlChars": "\\{^}%`]>[~<#|",
 "localCodePage": "IBM-1047",
 "remoteCodePage": "ISO8859-1",
 "authentication": {
```

1.1.2 Application Exploitation

1.1.2.1 GDKDLL31 Dynamic Load Library (DLL)

If you have a Language Environment C application you have been using to invoke the CDA APIs, and including the GDKCSS program object in the Binder (IEWL) step, you may now instead include the GDKDLL31 side file from SYS1.SIEASID. This will autocall the GDKDLL31 program object in

SYS1.SIEALNKE.

OLD:

Previous Binder Step linking with GDKCSS

```
// SET LMOD='''/OA65990/testbuff'''
//LINK     EXEC PGM=HEWL,
//  PARM='RMODE=ANY,DYNAM=DLL,ALIASES=NO,UPCASE=NO,MAP,XREF,RENT,CALL,C
//             ASE=MIXED'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(25,5))
//SYSLMOD  DD PATH=&LMOD,
//            PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXO,SIRWXG,SIRWXU)
//SYSLIB   DD  DSN='CEE.SCEELKEX',DISP=SHR
//         DD  DSN='CEE.SCEELKED',DISP=SHR
//         DD  DSN='CBC.SCCNOBJ',DISP=SHR
//         DD  DSN='SYS1.CSSLIB',DISP=SHR
//         DD  DSN='SYS1.LINKLIB.PDSE',DISP=SHR
//C8961    DD  DSN='CEE.SCEEOBJ',DISP=SHR
//SYSLIN   DD *
 INCLUDE SRCMOD
 INCLUDE SYSLIB(GDKCSS)
 AUTOCALL C8961
//SRCMOD   DD DISP=(OLD,DELETE),DSN=&&PGMOBJ
```

NEW:

```
// SET LMOD='''/OA65990/testbuff'''
//LINK     EXEC PGM=HEWL,
//  PARM='RMODE=ANY,DYNAM=DLL,ALIASES=NO,UPCASE=NO,MAP,XREF,RENT,CALL,C
//             ASE=MIXED'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(25,5))
//SYSLMOD  DD PATH=&LMOD,
//            PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXO,SIRWXG,SIRWXU)
//SYSLIB   DD  DSN='CEE.SCEELKEX',DISP=SHR
//         DD  DSN='CEE.SCEELKED',DISP=SHR
//         DD  DSN='CBC.SCCNOBJ',DISP=SHR
//         DD  DSN='SYS1.CSSLIB',DISP=SHR
//         DD  DSN='SYS1.LINKLIB.PDSE',DISP=SHR
//SIDEDECK DD  DSN='SYS1.SIEASID',DISP=SHR
//C8961    DD  DSN='CEE.SCEEOBJ',DISP=SHR
//SYSLIN   DD *
 INCLUDE SRCMOD
 INCLUDE SIDEDECK(GDKDLL31)
 AUTOCALL C8961
//SRCMOD   DD DISP=(OLD,DELETE),DSN=&&PGMOBJ
```

1.1.2.2 Target Different LOG Output

The optional parameter, logOutput can be specified, with a string value that is the z/OS UNIX absolute path name to a file, z/OS Data Set in the format //'<dsname>', or DD name in the format DD:<name>.

This name will be opened in append mode for the output.

1.1.2.3 GDK_SECONDARY CDA Symbol

In the metadata optional parameter, add a key name of your choosing followed by &GDK_SECONDARY. When using the GDK_PATH_DATALOCATION on a GDKWRITE call, and the dataLocation is the name of a z/OS Data Set, the secondary allocation of the data set will be formatted into the metadata tag for the object.

1.1.2.4 Avoid Read of Keyfile/Provider File

The optional parameter, keyFile, can be specified. The value should point to a buffer containing the contents of a gdkkeyf.json file for the current RACF user. Another optional parameter, providerFile, can be specified. The value should point to a buffer containing the contents of the requested provider file that should be used in the operation.

1.1.2.5 Change Text Conversion Defaults

The default conversion of text when requested, is from IBM-1047 locally, to ISO8859-1 on the remote system. The default conversion, or conversion default values found in the provider file, can be overridden by specifying the localCodePage, optional parameter. The value should have a codepage name that conforms to the names allowed for iconv() functions. The remote code page value may be overridden by specifying the remoteCodePage optional parameter. That value should have a codepage name that conforms to the names allowed for iconv() functions.

Link to the documentation where the codepage names are listed:

https://www.ibm.com/docs/en/zos/3.1.0?topic=cscu-universal-coded-character-set-converters

1.1.2.1 z/OS 3.1 AMODE64 JSON APIs

New APIs using JSON as input and output are made available. There is an AMODE31 version on z/OS v2r5 and above as well as an AMODE64 version on z/OS 3.1, depending on your application environment. Applications wanting to use Python or Java Native Interface should use the AMODE64 versions of the APIs. Your application should use the z/OS Binder, including the GDKDLL64 sidefile found in SYS1.SIEASID, as well as specifying the AMODE64 Binder parameter.

Example Binder JCL

```
// SET LMOD='''/usr/lib/mylib.so'''
//LINK     EXEC PGM=HEWL,
//  PARM='RMODE=ANY,DYNAM=DLL,ALIASES=NO,UPCASE=NO,MAP,XREF,RENT,CALL,C
//             ASE=MIXED,AMODE=64'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(TRK,(25,5))
//SYSLMOD  DD PATH=&LMOD,
//            PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXO,SIRWXG,SIRWXU)
//SYSLIB   DD  DSN='CEE.SCEELKEX',DISP=SHR
//         DD  DSN='CEE.SCEELKED',DISP=SHR
//         DD  DSN='CBC.SCCNOBJ',DISP=SHR
//         DD  DSN='SYS1.CSSLIB',DISP=SHR
//         DD  DSN='SYS1.LINKLIB.PDSE',DISP=SHR
//SIDEDECK DD  DSN='SYS1.SIEASID',DISP=SHR
//C8961    DD  DSN='CEE.SCEEOBJ',DISP=SHR
//SYSLIN   DD *
 INCLUDE SRCMOD
```

```
 INCLUDE SIDEDECK(GDKDLL64)
 AUTOCALL C8961
//SRCMOD   DD DISP=(OLD,DELETE),DSN=&&PGMOBJ
```

1.1.3 GDKUTIL New Functions

1.1.3.1 CONVERT Keyword Overrides

The UPLOAD and DOWNLOAD commands that process text files or data sets can now use the extended CONVERT keyword, which allows specification of any valid codepage name. The format is local code page name followed by an optional remote code page name. So, if you have a data set with the IBM-278 characters, and wanted to upload it as UTF-8 text, you would use the following keyword:

```
CONVERT(IBM-278,UTF-8)
```

The value names come from the list of names allowed on the iconv() function. A list can be found at:

https://www.ibm.com/docs/en/zos/3.1.0?topic=cscu-universal-coded-character-set-converters

1.1.3.2 Specifying a Long Object Name

If you have a long object name you want to specify on the JCL, you can use the new OBJNAMEX DD in place of the original OBJNAME DD. When OBJNAMEX exists in the job, every line/record of the DD will be read, trimming the leading and trailing whitespace, concatenating the remaining contents together to build the object name. Internal blank characters will not be trimmed. Sequence numbers at the end of the record will be considered as part of the name contents, so it is preferred not to use them. Simply change the OBJNAME DD to OBJNAMEX and you can use the new functionality.

Example JCL using a long object name

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 UPLOAD PROVIDER(CLOUD1) CONVERT(IBM-279)
/*
//LOCNAME  DD *
  HLQ1.APP.LOCAL.OUTPUT(0)
/*
//OBJNAMEX DD *
 /bank-loan-app
 /weekly
 /processing/2019-06-15/mid-year
 /cloud-app/results/
 2019-06-15.0915.04325.csv
/*
```

1.1.3.3 CREDENTIALS Command

A new command, CREDENTIALS is added. It expects one of the following sub-parameters: ADD, DELETE, or LIST. This command will use the provider name from the PROVIDER keyword, along with the optional BUCKET DD to perform the credentials management function.

CREDENTIALS(ADD)

This command will use the PROVIDER name, and the name specified on the BUCKET DD, along with the UNIX file name or data set name found in the CREDSNAM DD, to add credentials to the gdkkeyf.json

keyfile for the current user. The UNIX file or data set from the CREDSNAM DD will be read, expecting a valid JSON document containing key/value pairs for the credentials to be added.

If you are adding credentials for an S3 cloud provider, the JSON document would hold “accessKey”: “<my_new_S3_AccessKey>”, and “secretAccessKey”: “<my_new_S3_SecretKey>”. Where the values are your new access key ID and secret access key ID.

If you have saved a JSON credentials file from Google Cloud Platform locally on z/OS UNIX, the JSON document would hold: “credentials-file”: “/path/to/the/GCP/service_credentials.json” . Where the value is the absolute path to the saved GCP credentials file.

If you are adding credentials for an Azure cloud provider, the JSON document would hold “accessKey”: “<my_storage_account_name>”, “secretAccessKey”: “<my_new_Azure_key>” . Where <my_storage_account_name> is the name of your Azure storage account, and <my_new_Azure_key> is the new key from the Azure control panel.

If the userid associated with the JCL JOB is not the userid you want to save the credentials for, you can use the new ZUSERID keyword to specify which z/OS userid the credentials are to be saved for. The JCL JOB userid must have write permission to the other z/OS userid ~/gdk/gdkkeyf.json file. The JCL JOB userid must also have ALTER permission to the CSFKEYS class for the GDK.<ZUSERID>.<PROVIDER> profile so that the new encryption key can be saved under that ICSF keylabel. The syntax is: ZUSERID(<USERID>).

The BUCKET DD is optionally examined for a bucket name that the credentials apply solely to. If not specified in the JCL, then the default (all buckets) of / (forward slash) is used.

The example below shows S3 credentials being added for the CLOUD1 provider for the default (all buckets).

Example JCL to save new default credentials for the CLOUD1 provider

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(ADD) PROVIDER(CLOUD1)
/*
//CREDSNAM DD *
 /u/user1/add_S3_creds.json
/*
```

Contents of /u/user1/add_S3_creds.json

```
{
  "accessKey": "S3Cloud_access_key",
  "secretAccessKey": "S3Cloud_secret_key"
}
```

CREDENTIALS(DELETE)

If you want to delete some credentials from the keyfile, you can use the CREDENTIALS(DELETE) command. This command will use the value from the PROVIDER keyword, along with the optional BUCKET DD statement contents to delete the credentials for the provider/bucket combination. If the BUCKET DD statement doesn’t exist, the default (all buckets) / (forward slash) will be used.

If the userid associated with the JCL JOB is not the userid you want to save the credentials for, you can use the new ZUSERID keyword to specify which z/OS userid the credentials are to be saved for. The JCL JOB userid must have write permission to the other z/OS userid ~/gdk/gdkkeyf.json file. The JCL JOB

userid must also have ALTER permission to the CSFKEYS class for the GDK.<ZUSERID>.<PROVIDER> profile so that the new encryption key can be saved under that ICSF keylabel. The syntax is: ZUSERID(<USERID>).

Example JCL to delete default credentials for the CLOUD1 provider.

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(DELETE) PROVIDER(CLOUD1)
/*
```

CREDENTIALS(LIST)

If you wish to see what buckets there are currently credentials saved for, you can use the CREDENTIALS(LIST) command, specifying a cloud provider name. This command will return a list of the bucket names that credentials exist for in the keyfile. The PROVIDER keyword may specify a specific cloud name, or an asterisk (*). When a specific cloud name is specified, the SYSPRINT output contains a table of each bucket name with the timestamp that credential was saved. When the asterisk (*) is specified, all entries for all cloud providers is returned. The SYSPRINT output will contain a table of each bucket name with the cloud provider name that credential is associated with.

If the userid associated with the JCL JOB is not the userid you want to save the credentials for, you can use the new ZUSERID keyword to specify which z/OS userid the credentials are to be saved for. The JCL JOB userid must have write permission to the other z/OS userid ~/gdk/gdkkeyf.json file. The JCL JOB userid must also have ALTER permission to the CSFKEYS class for the GDK.<ZUSERID>.<PROVIDER> profile so that the new encryption key can be saved under that ICSF keylabel. The syntax is: ZUSERID(<USERID>).

Example JCL to list the credential entries for CLOUD1

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(LIST) PROVIDER(CLOUD1)
/*
```

Example JCL to list all credential entries for all providers for a different user

```
//EXEC     PGM=GDKUTIL,REGION=0M,USERID=USER1
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(LIST) PROVIDER(*) ZUSERID(USER256)
/*
```

2 Publication Updates

2.1 MVS Programming: Callable Services for High Level Languages

MVS Programming: Callable Services for High Level Languages

SA23-1377

Chapter 29: Syntax, linkage, and programming considerations is updated under the Linkage considerations as follows:

Add a new paragraph after the **Linkage stub method**, and before the **Direct linkage method** as follows:
Dynamic Link Library

> (Recommended for AMODE31 C language callers) Use the GDKDLL31 side file found in SYS1.SIEASID when binding your object code. GDKDLL31 assumes that you are already in a Language Environment in AMODE 31, and the calling language is XLC. At runtime, the GDKDLL31 dynamic load library found in SYS1.SIEALNKE will be found and used by the appropriate API. When Binding your C code into a program, ensure that DYNAM=DLL is specified as a IEWL parameter.
>
> (Recommended for AMODE64 C language callers) Use the GDKDLL64 side file found in SYS1.SIEASID when binding your object code. GDKDLL64 assumes that you are already in a Language Environment in AMODE 64, and the calling language is XLC. At runtime, the GDKDLL64 dynamic load library found in SYS1.SIEALNKE will be found and used by the appropriate API. When Binding your C code into a program, ensure that DYNAM=DLL and AMODE=64 is specified as a IEWL parameter.

In Chapter 30, the optional parameter table for all APIs is updated to add the following row.

Name	Type	Description
logOutput	Address	Specifies the address of a null terminated string that is the name of the output for CDA log messages. It must conform to the naming convention documented for the fopen() function. e.g. DD:OUTDD, /UNIX/file/name.txt, or //'IBMUSER.CDALOG.OUTPUT' . When the log-level is not NONE, CDA logging messages will be written to the requested output name, which can be a DD, z/OS UNIX file, or data set. The userid invoking the API must have enough permission to append data to the output target. If not specified, it defaults to stderr, which in a JCL environment is directed to a DD named SYSOUT.
providerFile	Address	Specifies the address of a null terminated string containing a valid JSON document with all of the provider file contents describing how to communicate with the target cloud object server. When this optional parameter is specified, CDA will copy the contents and use that as the provider specification and not attempt to read the provider file from the user's provider directory or the system default provider directory.
keyFile	Address	Specifies the address of a null terminated string containing a valid JSON document consisting of the contents of a CDA keyfile for the user. When this optional parameter is specified, CDA will copy the contents and use that as the keyfile contents to be used when authenticating with the target cloud object server.
localCodePage	Address	Specifies the address of a null terminated string containing a name of a valid code page to be used in conversion of local text data from local to remote, or remote to local.
remoteCodePage	Address	Specifies the address of a null terminated string containing a name of a valid code page to be used in conversion of remote

		text data from local to remote, or remote to local.

In Chapter 30, the table containing the list of CDA Symbols for the GDKWRITE API is updated to add the following row for GDK_SECONDARY:

CDA symbol	Meaning
GDK_SECONDARY	Secondary allocation amount for the z/OS data set in the format: <nnn>-<unit>, where *nnn* is the number or amount, and unit is the allocation unit. e.g. CYL, TRK, BLK, REC, MB, KB, B

In Chapter 30, Add the following row to the Optional Parameters table for the GDKINIT API:

Name	Type	Description
multi_thread	Address	Specifies the address of a null terminated string that indicates that this CDA session may be performed in parallel with other CDA sessions. Specifying this to be “true” will cause CDA to use the ICSF APIs CSNPTRC, CSNPHMG, and CSNPTRD to compute the SHA-256 HMAC instead of the ICSF PKCS#11 APIs.

In Chapter 30, add new sections for the following APIs:

2.1.1.1 gdkkeysrJ/gdkkeysr64J – Retrieve cloud credentials JSON API

gdkkeysrJ/gdkkeysr64J – Retrieve cloud credentials JSON API

The gdkkeysrJ C API is used to retrieve the stored cloud credentials for a user associated with a cloud provider and resource with the input and output being JSON.

Description

The gdkkeysrJ C API is used to retrieve the cloud credentials for the specified cloud provider and resource. When called, the key file for the user will be read and the encrypted cloud credentials matching the request will be decrypted and returned. An AMODE64 version of the C API is available on z/OS 3.1 and called with gdkkeysr64J. The gdkkeysrJ C API processing can additionally be modified through optional parameters.

Syntax

```
char* gdkkeysrJ(char *__JSON_in)

char* gdkkeysr64J(char *__JSON_in)
```

Input Parameter

The input parameter is a null-terminated character string JSON document containing the following key/value pairs:

provider

Specifies the name of the cloud provider the credentials are associated with. Required.

resource

Specifies the bucket name the credentials are associated with. Optional. If not specified, the credentials associated with the default bucket / are returned.

maxKeyLen

Specifies a number that is the maximum length of each credential returned. Optional. Defaults to 8192 bytes.

optionalParms

Specifies a JSON object containing key/value pairs used to modify processing of the API.

UserID - RACF UserID used to retrieve cloud credentials.

Use-Config-File - **false** means that the config.json file should not be read for default configuration values.

Any other values mean that the config.json should be used.

log-level - The logging level can be set as desired. This value will override any default, or value in the config.json file. Logging messages are written to stderr unless overridden by the logOutput optional parameter. The levels in order of low to high severity are listed:

DEBUG means all logging messages are written.

INFO means only INFO and higher severity logging messages are written.

NOTICE means only NOTICE and higher severity logging messages are written.

WARNING means only WARNING and higher severity logging messages are written.

ERROR means only ERROR logging messages are written.

NONE means no messages are written regardless of severity.

logOutput

Specifies the name a location where the logging output messages should be written. It must conform to the naming convention documented for the fopen() function. e.g. DD:OUTDD, /UNIX/file/name.txt, or //'IBMUSER.CDALOG.OUTPUT' . When the log-level is not NONE, CDA logging messages will be written to the requested output name, which can be a DD, z/OS UNIX file, or data set. The userid invoking the API must have enough permission to append data to the output target. If not specified, it defaults to stderr, which in a JCL environment is directed to a DD named SYSOUT.

Example input JSON to retrieve the default credentials for the IBMCOS cloud provider

```
{
  "provider": "IBMCOS",
  "optionalParms":{
    "Use-Config-File": "false",
    "log-level": "DEBUG"
  }
}
```

Return JSON

The return string is a JSON containing the following fields as appropriate. It is the responsibility of the caller of the API to free the returned memory.

rc

CDA return code from processing

provider

The name of the cloud provider the credentials are associated with.

resource

The bucket name the credentials are associated with.

accessKey

Access Key ID value if present in the key file.

secretAccessKey

Secret Access Key value if present in the key file.

tenant

Tenant value if present in the key file.

userid

Userid value if present in the key file.

password

Password value if present in the key file.

accessURL

Access URL value if present in the key file.

maxKeyLen

Specifies the maximum key buffer length to use when retrieving the credentials. If not specified, the default size is 8192 bytes.

messages

JSON array of output messages from processing.

Example output JSON

```
{
  "rc": 0,
  "provider": "IBMCOS",
  "resource": "/bucket1",
  "accessKey": "myExampleAccessKeyID1",
  "secretAccessKey": "myExampleSecretAccessKey1",
  "accessURL": https://loc.site.org/auth/v1.0,
  "messages": [
    "Success"
  ]
}
```

2.1.1.2 gdkkeyadJ/gdkkeyad64J – Store cloud credentials JSON API

gdkkeyadJ/gdkkeyad64J – Store cloud credentials JSON API

The gdkkeyadJ C API is used to store cloud credentials for a user associated with a cloud provider and resource with the input and output being JSON.

Description

The gdkkeyadJ C API is used to store the cloud credentials for the specified cloud provider and resource. When called, the passed cloud credentials are encrypted by ICSF, using a newly generated AES256 data key. The new data key is stored with ICSF under a keylabel of:

`GDK.<userid>.<provider>.<identifier>`

where:

- userid is the current RACF user's ID
- provider is the specified cloud provider in the user's ~/gdk/providers/ directory or system default directory off /usr/lpp/dfsms/gdk/providers/.
- identifier is a CDA used identifier used internally

For more information, see Chapter 27, “Cloud Data Access cloud credential storage,” on page xxx.

Storing the encrypted credentials. The keyfile in the current user's (or UserID optional parm) ~/gdk/ directory is read, and a new entry is created, or an existing entry is overwritten for the specified cloud provider and resource. An AMODE64 version of the C API is available on z/OS 3.1 and is called with gdkkeyad64J.The gdkkeyadJ C API processing can additionally be modified through optional parameters.

Syntax

```
char* gdkkeyadJ(char *__JSON_in)

char* gdkkeyad64J(char *__JSON_in)
```

Input Parameter

The input parameter is a null-terminated character string JSON document containing the following key/value pairs:

provider

Specifies the name of the cloud provider the credentials are associated with. Required.

resource

Specifies the bucket name the credentials are associated with. Optional. If not specified, the credentials associated with the default bucket, /, are returned.

accessKey

Access Key ID value for the credentials. Optional.

secretAccessKey

Secret Access Key value for the credentials. Optional.

tenant

Tenant value for the credentials. Optional.

userid

Userid value for the credentials. Optional.

password

Password value for the credentials. Optional.

accessURL

Access URL value for the credentials. Optional.

optionalParms

Specifies a JSON object containing key/value pairs used to modify processing of the API.

UserID - RACF UserID the cloud credentials are associated with.

Use-Config-File - **false** means that the config.json file should not be read for default configuration values. Any other values mean that the config.json should be used.

log-level - The logging level can be set as desired. This value will override any default, or value in the config.json file. Logging messages are written to stderr unless overridden by the logOutput optional parameter. The levels in order of low to high severity are listed:

DEBUG means all logging messages are written.

INFO means only INFO and higher severity logging messages are written.

NOTICE means only NOTICE and higher severity logging messages are written.

WARNING means only WARNING and higher severity logging messages are written.

ERROR means only ERROR logging messages are written.

NONE means no messages are written regardless of severity.

logOutput

Specifies the name a location where the logging output messages should be written. It must conform to the naming convention documented for the fopen() function. e.g. DD:OUTDD, /UNIX/file/name.txt, or //'IBMUSER.CDALOG.OUTPUT' . When the log-level is not NONE, CDA logging messages will be written to the requested output name, which can be a DD, z/OS UNIX file, or data set. The userid invoking the API must have enough permission to append data to the output target. If not specified, it defaults to stderr, which in a JCL environment is directed to a DD named SYSOUT.

Example input JSON to store the default credentials for an S3 cloud provider named IBMCOS

```
{
  "provider": "IBMCOS",
  "accessKey": "myAccessKey",
  "secretAccessKey": "mySecretKeyID",
  "optionalParms":{
    "Use-Config-File": "false",
    "log-level": "DEBUG"
  }
}
```

Return JSON

The return string is a JSON containing the following fields as appropriate. It is the responsibility of the

caller of the API to free the returned memory.

rc

CDA return code from processing

provider

The name of the cloud provider the credentials were associated with.

resource

The bucket name the credentials were associated with.

messages

JSON array of output messages from processing.

Example output JSON

```
{
  "rc": 0,
  "provider": "IBMCOS",
  "resource": "/",
}
```

2.1.1.3 gdkkeydlJ/gdkkeydl64J – Delete cloud credentials JSON API

gdkkeydlJ/gdkkeydl64J – Delete cloud credentials JSON API

The gdkkeydlJ C API is used to delete the stored cloud credentials for a user associated with a cloud provider and resource with the input and output being JSON.

Description

The gdkkeydlJ C API is used to delete the cloud credentials for the specified cloud provider and resource. When called, the key file for the user will be read and the encrypted cloud credentials matching the request will be decrypted and returned. An AMODE64 version of the C API is available on z/OS 3.1 and is called with gdkkeydl64J.The gdkkeydlJ C API processing can additionally be modified through optional parameters.

Syntax

```
char* gdkkeydlJ(char *__JSON_in)
char* gdkkeydl64J(char *__JSON_in)
```

Input Parameter

The input parameter is a null-terminated character string JSON document containing the following key/value pairs:

provider

Specifies the name of the cloud provider the credentials are associated with. Required.

resource

Specifies the bucket name the credentials are associated with. Optional. If not specified, the credentials associated with the default bucket / are returned.

optionalParms

Specifies a JSON object containing key/value pairs used to modify processing of the API.

UserID - RACF UserID used to find the cloud credentials to delete.

Use-Config-File - **false** means that the config.json file should not be read for default configuration values. Any other values mean that the config.json should be used.

log-level - The logging level can be set as desired. This value will override any default, or value in the config.json file. Logging messages are written to stderr unless overridden by the logOutput optional parameter. The levels in order of low to high severity are listed:

DEBUG means all logging messages are written.

INFO means only INFO and higher severity logging messages are written.

NOTICE means only NOTICE and higher severity logging messages are written.

WARNING means only WARNING and higher severity logging messages are written.

ERROR means only ERROR logging messages are written.

NONE means no messages are written regardless of severity.

logOutput

Specifies the name a location where the logging output messages should be written. It must conform to the naming convention documented for the fopen() function. e.g. DD:OUTDD, /UNIX/file/name.txt, or //'IBMUSER.CDALOG.OUTPUT' . When the log-level is not NONE, CDA logging messages will be written to the requested output name, which can be a DD, z/OS UNIX file, or data set. The userid invoking the API must have enough permission to append data to the output target. If not specified, it defaults to stderr, which in a JCL environment is directed to a DD named SYSOUT.

Example input JSON to delete the default credentials for the IBMCOS provider

```
{
  "provider": "IBMCOS",
  "optionalParms":{
    "Use-Config-File": "false",
    "log-level": "DEBUG"
  }
}
```

Return JSON

The return string is a JSON containing the following fields as appropriate. It is the responsibility of the caller of the API to free the returned memory.

rc

CDA return code from processing

provider

The name of the cloud provider the credentials are associated with.

resource

The bucket name the credentials are associated with.

accessKey

Access Key ID value if present in the key file.

secretAccessKey

Secret Access Key value if present in the key file.

tenant

Tenant value if present in the key file.

userid

Userid value if present in the key file.

password

Password value if present in the key file.

accessURL

Access URL value if present in the key file.

maxKeyLen

Specifies the maximum key buffer length to use when retrieving the credentials. If not specified, the default size is 8192 bytes.

messages

JSON array of output messages from processing.

Example output JSON

```
{
  "rc": 0,
  "provider": "IBMCOS",
  "resource": "/bucket1",
  "accessKey": "myExampleAccessKeyID1",
  "secretAccessKey": "myExampleSecretAccessKey1",
  "accessURL": https://loc.site.org/auth/v1.0,
  "messages": [
    "Success"
  ]
}
```

2.1.1.4 gdkkeygrJ/gdkkeygr64J – List resources for provider JSON API

gdkkeygrJ/gdkkeygr64J – List cloud credentials resources JSON API

The gdkkeygrJ C API is used to retrieve a list of cloud credentials identified by their bucket names or resources for a user associated with a cloud provider with the input and output being JSON.

Description

The gdkkeygrJ C API is used to retrieve a list of the cloud credentials for the specified cloud provider. When called, the key file for the user will be read and a list containing the bucket name or resource will be returned. This API only identifies which credentials exist. No credentials are returned. An AMODE64 version of the C API is available on z/OS 3.1 and is called with gdkkeygr64J. The gdkkeygrJ C API processing can additionally be modified through optional parameters.

Syntax

```
char* gdkkeygrJ(char *__JSON_in)
char* gdkkeygr64J(char *__JSON_in)
```

Input Parameter

The input parameter is a null-terminated character string JSON document containing the following key/value pairs:

provider

Specifies the name of the cloud provider the credentials are associated with. Required. This may be the special character, asterisk (*), to request a list of credentials for all cloud providers.

optionalParms

Specifies a JSON object containing key/value pairs used to modify processing of the API.

UserID - RACF UserID to list the cloud credentials for. The current user must have read access to the home directory ~/gdk/gdkkeyf.json file.

Use-Config-File - **false** means that the config.json file should not be read for default configuration values. Any other values mean that the config.json should be used.

log-level - The logging level can be set as desired. This value will override any default, or value in the config.json file. Logging messages are written to stderr unless overridden by the logOutput optional parameter. The levels in order of low to high severity are listed:

DEBUG means all logging messages are written.

INFO means only INFO and higher severity logging messages are written.

NOTICE means only NOTICE and higher severity logging messages are written.

WARNING means only WARNING and higher severity logging messages are written.

ERROR means only ERROR logging messages are written.

NONE means no messages are written regardless of severity.

logOutput

Specifies the name a location where the logging output messages should be written. It must conform to the naming convention documented for the fopen() function. e.g. DD:OUTDD, /UNIX/file/name.txt, or //'IBMUSER.CDALOG.OUTPUT' . When the log-level is not NONE, CDA logging messages will be written to the requested output name, which can be a DD, z/OS UNIX file, or data set. The userid invoking the API must have enough permission to append data to the output target. If not specified, it defaults to stderr, which in a JCL environment is directed to a DD named SYSOUT.

Example input JSON to retrieve the default cloud credentials for the IBMCOS cloud provider

```
{
  "provider": "IBMCOS",
  "optionalParms":{
    "Use-Config-File": "false",
    "log-level": "DEBUG"
  }
}
```

Return JSON

The return string is a JSON containing the following fields as appropriate. It is the responsibility of the caller of the API to free the returned memory.

rc

CDA return code from processing

provider

The name of the cloud provider the credentials are saved. Only returned when the 'All providers' special value of asterisk (*) is requested.

resource

The bucket name that has credentials saved.

timestamp

The date and time the credentials were saved in the key file.

messages

JSON array of output messages from processing.

Example output JSON

```
{
  "rc": 0,
  "provider": "IBMCOS",
  "results": [
    { "resource": "/bucket", "timestamp": "2024-09-26 14:43:17" },
    { "resource": "/", "timestamp": "2024-09-27 17:07:14" },
    { "resource": "/testbucket", "timestamp": "2024-09-27 17:57:40" }
  ]
}
```

2.2 DFSMSdfp Utilities

DFSMSdfp Utilities

SC23-6864

Chapter 2: GDKUTIL (Cloud Object Utility) Program, is updated.

The Job Control table is updated to add another row as follows:

Statement	Use
CREDSNAM	Defines the name of a UNIX file or z/OS data set that contains the credentials JSON to be used in the CREDENTIALS(ADD) command. It can be in-stream data (DD*) that identifies a z/OS UNIX file absolute path, or z/OS data set name. Or it can identify a z/OS UNIX PATH or Data Set that contains the z/OS UNIX file absolute path or z/OS Data Set name.
OBJNAMEX	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 if OBJNAME DD is not specified)

After the description of the BUCKET statement, add the following description for the CREDSNAM statement.

CREDSNAM Statement

The CREDSNAM statement identifies the location of JSON that contains the credentials to be used by the CREDENTIALS(ADD) command. The location may be a z/OS UNIX absolute path, or a Data Set. The DD will be opened and all records/lines are read, concatenating them together after trimming leading and trailing whitespace to create the z/OS data set or UNIX file to open.

The JSON document may contain the following keys with your credential values as needed:

- accessKey – The value is the S3 HMAC Access Key ID used in the authentication.
- secretAccessKey – The value is the S3 HMAC Secret Access Key used for authentication.
- tenant – The tenant value used in the authentication.
- userid – The userid value used for authentication.
- password – The password value used for authentication.
- accessURL – The Access URL used during authentication.
- credentials-file – The absolute path to the UNIX file that contains the JSON credentials obtained from the cloud provider Note: this is only used with OAUTH_2 authentication models. This value may override a specified accessURL. Both may not be specified in a JSON credentials document.

Note:

The first occurrence of each key is the value used. Subsequent duplicate keys are ignored.

Example JSON credentials found in /u/user1/tempcreds.json

```
{
  "accessKey": "myIBM_Cloud_AccessKey",
  "secretAccessKey": "myIBM_Cloud_SecretAccessKey"
}
```

Example JSON credentials found in /u/user/temp_credentials.json

```
{
  "credentials-file": "/u/ibmuser/GCPcreds/credentials.json"
}
```

After the description of the OBJNAME statement, add the following description for the OBJNAMEX statement:

OBJNAMEX Statement

The OBJNAMEX statement describes the location that holds the object name in Cloud Storage that will be used in the utility operation. All records in the data set, member, or in-stream data, or lines in a UNIX file is read and the content is used as the object name. Leading and trailing whitespace is ignored when building the object name. Note that sequence numbers will be considered as part of the record and included in the text.

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 regarding object names.

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 ("|")

The OBJNAMEX DD may contain simply a bucket name. A bucket name is defined to be a name that starts with a forward slash character, and ends with a forward slash character with no other forward slash characters in between. The LIST command will return a list of objects found within a bucket in this case.

If the OBJNAMEX is simply the forward-slash character, and the LIST command is used, then a list of the accessible buckets is displayed. The PREFIX keyword may also adjust the object or bucket names that are retrieved.

In the SYSIN Statement table, make the following changes:

In the Commands section, add the following row after the OPERATION command row:

CREDENTIALS(ADD|DELETE|LIST)	Requests Credential management functions: ADD – Add the cloud credentials found in the JSON document found in the z/OS UNIX file or data set named by the CREDSNAM DD statement. The credentials will be saved for the named PROVIDER name. The optional BUCKET DD statement may be used to describe the

	bucket name the credentials apply to. If the BUCKET DD is not provided, the credentials will be saved for the generic / bucket, meaning all buckets. If credentials already exist for the provider, they will be deleted before the new credentials are added. DELETE – Delete the saved credentials for the named PROVIDER and bucket name. If the optional BUCKET DD wasn't specified to indicate the specific bucket name for the credentials, the generic / bucket is defaulted to. LIST – Display a list of credentials (by bucket name) for the PROVIDER specified. A provider name of asterisk (*) may be specified to indicate credentials for all providers should be displayed.

The CONVERT keyword description in the SYSIN table is updated to be specific about IBM-1047 as the EBCDIC default as follows:

Keyword	Use
CONVERT	Optional: Indicates that the data should be converted from EBCDIC (IBM-1047) to ASCII (ISO8859-1) on UPLOAD, or from ASCII (ISO8869-1) to EBCDIC (IBM-1047) 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.

New keywords are also added to the SYSIN table.

Keyword	Use
CONVERT(<localCCSID>{,<remoteCCSID>})	Optional: Indicates that DFSMSdfp CDA should convert the data the local CCSID (code page) to the remote CCSID (code page) on UPLOAD, or from the remote code page to the local code page on DOWNLOAD. The remote CCSID is optional, and will default to ASCII (ISO8859-1) if not specified. Minimum text CONV(<localCCSID>)
ZUSERID(<user>)	Used with the CREDENTIALS command to indicate that the credentials management request should be performed for the named user. For ADD and DELETE, the user ID associated with the JOB must have ALTER access to the <user>'s GDK.<user> keylabel in the CSFKEYS class. Additionally, the JOB userid must have write access to the <user>'s ~/gdk/gdkkeyf.json key file in the <user>'s home directory. For LIST requests, the user ID associated with the JOB must have read access to the <user>'s ~/gdk/gdkkeyf.json

	file.

In the CDA Variables table under the SYSIN section to add the following row for GDK_SECONDARY:

CDA symbol	Meaning
GDK_SECONDARY	Secondary allocation amount for the z/OS data set in the format: <nnn>-<unit>, where *nnn* is the number or amount, and unit is the allocation unit. e.g. CYL, TRK, BLK, REC, MB, KB, B

Add the following examples to the GDKUTIL Examples section:

Example 9: Save new S3 cloud credentials

The example below shows the default S3 credentials being added for the CLOUD1 provider. (Default means all buckets). The S3 credentials are found in a separate UNIX file with the absolute path: /u/user1/add_S3_creds.json. The content of the UNIX credentials file is a JSON document as displayed below.

Example JCL to save new default credentials for the CLOUD1 provider

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(ADD) PROVIDER(CLOUD1)
/*
//CREDSNAM DD *
 /u/user1/add_S3_creds.json
/*
```

Contents of /u/user1/add_S3_creds.json

```
{
  "accessKey": "S3Cloud_access_key",
  "secretAccessKey": "S3Cloud_secret_key"
}
```

Example 10. Delete Credentials

The example below shows the JCL used to delete the default credentials entry for the CLOUD1 provider.

Example JCL to delete default credentials for the CLOUD1 provider.

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(DELETE) PROVIDER(CLOUD1)
/*
```

Example 11. List Credential Entries

The first example below shows the JCL used to list the credential entries for the CLOUD1 provider for the current JOB userid.

The second example below shows the JCL used to list the credential entries for all of the providers for the z/OS userid: USER256.

Example JCL to list the credential entries for CLOUD1

```
//EXEC     PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(LIST) PROVIDER(CLOUD1)
/*
```

Example JCL to list all credential entries for all providers for a different user

```
//EXEC     PGM=GDKUTIL,REGION=0M,USERID=USER1
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 CREDENTIALS(LIST) PROVIDER(*) ZUSERID(USER256)
/*
```

2.3 MVS System Messages Volume 5 (EDG – GLZ)

SA38-0672-60

Chapter 10. GDK Messages is updated to add the following messages.

2.3.1 GDKU0027E CANNOT SPECIFY <ddname1> AND <ddname2>

GDKU0027E CANNOT SPECIFY <ddname1> AND <ddname2>

Explanation

<ddname1> and <ddname2> were both specified on the JCL for the task. However, both DD statements relate to the same information and the GDKUTIL utility cannot choose which information was intended.

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

Operator response
None

System Programmer response
Examine the named DD statements and remove the unnecessary DD statement.

Source
DFSMSdfp CDA

2.3.2 GDKU0028E <ddname> DD CONTAINS TEXT LONGER THAN <length>

GDKU0028E <ddname> DD CONTAINS TEXT LONGER THAN <length>

Explanation
The specified DD <ddname> contained text longer than allowed. Max length is <length>.

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

Operator response
None.

System Programmer response
If the DD is supposed to contain an object name, ensure that the text is not too long.

Source
DFSMSdfp CDA

2.3.3 GDKU0029E CREDS(ADD) REQUIRES CREDSNAM DD TO EXIST WITH A DATA SET NAME OR UNIX PATH NAME

GDKU0029E CREDS(ADD) REQUIRES CREDSNAM DD TO EXIST WITH A DATA SET NAME OR UNIX PATH NAME

Explanation
The CREDENTIALS(ADD) command was specified, but the CREDSNAM DD statement was not specified for the STEP with a z/OS UNIX path name or data set name.

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

Operator response
None.

System Programmer response
DFSMSdfp CDA requires the input file containing a JSON document with the credentials to save.

Source
DFSMSdfp CDA

2.3.4 GDKU0030I BUCKET NAME NOT SPECIFIED FOR CREDS(ADD). CREDENTIALS WILL APPLY TO ALL BUCKETS.

GDKU0030I BUCKET NAME NOT SPECIFIED FOR CREDS(ADD). CREDENTIALS WILL APPLY TO ALL BUCKETS.

Explanation
The BUCKET or OBJNAME DD statement was not specified to indicate a specific bucket that the credentials should apply to. The credentials will be used for all buckets unless credentials for a specific bucket have been saved.

System action
The Task continues.

Operator response
None.

System Programmer response
None.

Source
DFSMSdfp CDA

2.3.5 GDKU0031E ERROR READING CREDENTIALS JSON: <error_desc>

GDKU031E When reading the credentials file named on the CREDSNAM DD, an error occurred while reading the contents.

Explanation
When reading the credentials file named on the CREDSNAM DD, an error occurred while reading the contents.

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

Operator response
None.

System Programmer response
Examine the file named on the credentials file for reasons it cannot be read.

Source
DFSMSdfp CDA

2.3.6 GDKU0032E UNABLE TO OPEN CREDENTIALS JSON: <error_desc>

GDKU0032E UNABLE TO OPEN CREDENTIALS JSON: <error_desc>

Explanation
An error occurred when trying to open the file named on the CREDSNAM DD.

System action

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

Operator response
None.

System Programmer response
Use the <error_desc> to identify why the named credentials JSON file couldn't be opened.

Source
DFSMSdfp CDA

2.3.7 GDKU0033E ERROR INITIALIZING JSON PARSER. RC: <rc> RSN: <rsn> DIAG: <diagnostic_info>

GDKU0033E ERROR INITIALIZING JSON PARSER. RC: <rc> RSN: <rsn> DIAG: <diagnostic_info>

Explanation
When preparing to parse the JSON from the credentials JSON file, and error occurred with the HWTJINIT call.

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

Operator response
None.

System Programmer response
Use the Return and Reason codes, along with the diagnostic information to determine the error encountered with HWTJINIT.

Source
DFSMSdfp CDA

2.3.8 GDKU0034E ERROR PARSING CREDENTIALS FILE. RC: <rc> RSN: <rsn> DIAG: <diagnostic_info>

GDKU0034E ERROR PARSING CREDENTIALS FILE. RC: <rc> RSN: <rsn> DIAG: <diagnostic_info>

Explanation
An error occurred while using HWTJPARS to parse the JSON credentials file.

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

Operator response
None.

System Programmer response
Use the Return and Reason codes, along with the diagnostic information to determine the error encountered with HWTJPARS. Examin the JSON for correctness. Extraneous commas are often an issue.

Source
DFSMSdfp CDA

2.3.9 GDKU0035W UNABLE TO GET VALUE FOR: <key> KEY. RC: <rc> RSN: <rsn> DIAG: <diagnostic_info>

GDKU0035W UNABLE TO GET VALUE FOR: <key> KEY. RC: <rc> RSN: <rsn> DIAG: <diagnostic_info>

Explanation
An error occured while using HWTJGVAL to get the value for the named key.

System action
The Task continues.

Operator response
None.

System Programmer response
The HWTJGVAL API returned an error while attempting to retrieve the value for the <key> key. Use the return and reason codes, along with the diagnostic information to determine the cause of the issue.

Source
DFSMSdfp CDA

2.3.10 GDKU0116I LIST OF CREDENTIALS FOR PROVIDER <provider>

GDKU0116I LIST OF CREDENTIALS FOR PROVIDER <provider>

Explanation
A CREDENTIALS(LIST) command was requested. The following lines display the bucket that credentials are saved for as well as a time stamp when the credentials were saved. When asterisk (*) is used for the provider name, the provider name is displayed instead.

System action
The Task continues.

Operator response
None.

System Programmer response
None.

Source
DFSMSdfp CDA