# OA65224 Publication Updates

DFSMSdfp CDA

04/01/2024

Andrew Wilt

This document describes the updates to the z/OS 3.1 publications as a result of the new function delivered via OA65224.

## 1 Overview

The OA65224 APAR delivers support for new CDA APIs; GDKINIT, GDKTERM, GDKGEN, GDKVALD, and GDKQUERY.

- GDKINIT/GDKTERM together introduce support for a 'CDA session' allowing for multiple CDA API calls over a single connection.
- GDKGEN allows a caller to request execution of a named operation that is detailed in the provider file, such as CREATEBUCKET.
- GDKVALD allows a caller to validate a provider file by requesting checks on operations, keys, as well as the ability to retrieve the value associated with a particular key.
- GDKQUERY allows a caller to query the functionality supported by the DFSMSdfp CDA code currently on the z/OS system.

The GDKWRITE and GDKGET APIs are updated for the GDK_EXIT_DATALOCATION source type to utilize the WRITELARGEOBJECT and GETLARGEOBJECT operations in a provider file.

The GDKLIST API is updated to support prefix, delimiter, and marker url parameters.

Additionally, the GDKUTIL program is updated to support a new OPER(<operation>) command as well as updates to the LIST command functionality:

- List all buckets with a common PREFIX.
- List objects with a common PREFIX
- List objects with a delimiter (DELIM)
- UPLOAD multiple objects with one command
- DOWNLOAD multiple objects with one command
- DELETE multiple objects with one command

### 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

The new functionality is enabled by new sections and information found in the provider file for the cloud store. The CDA sample provider files found in /usr/lpp/dfsms/gdk/samples/providers/ are updated to reflect the enablement of the new functions. You will need to update the appropriate provider file in the user's home directory ~/gdk/providers/ to include the new statements.

One way to find the updates is to compare your provider file with the equivalent sample provider file. Another way is to take the sample provider file as-is, and update it with the unique values for your cloud provider. These keys are often: host, port, sslCiphers, encodeChars and other keys in the first section of the JSON.

Following is a description of the updates and their associated provider file sections.

- **Create a bucket** - CREATEBUCKET is a new operation added to the supportedOperations array. Add this entire section to the supportedOperations array. The IBMCOS.json sample additionally includes the MESSAGE_BODY needed to specify a specific LocationConstraint which the IBM Cloud calls a storage class. (us-smart in this case) https://cloud.ibm.com/docs/cloud-object-storage?topic=cloud-object-storage-classes
- **Delete a bucke**t - DELETEBUCKET is a new operation added to the supportedOperations array. Add this entire JSON object to the supportedOperations array. This operation sends a delete bucket request. It may fail if the bucket is not empty of objects.
- **Create a bucket with S3 object lock -** CREATEOBJLOCKBUCKET is a new operation added to the AWS4 model sample provider files, IBMCOS.json and S3CLOUD.json, that describes creating a bucket that has object lock enabled.
- **List Objects with a common prefix** – In the LISTOBJECT operation in AWS4 provider files, a new JSON object is added to the requestParameters array. This object has a mechanism of URL_PARM, and a descripter of “prefix=<GDK_PREFIX>”.
- **List Objects with names up to a delimiter value**. In the LISTOBJECT operation in AWS4 provider files, a new JSON object is added to the requestParameters array. The object uses the URL_PARM mechanism with a descriptor of “delimiter=<GDK_DELIMITER>”. Additionally, a new JSON object needs to exist in the responseResults array with a mechanism of DELIMITER_MESSAGE_BODY. This describes to CDA how to parse the response results with delimiter was requested.
- **Ensure all object names are returned for LISTOBJECT requests**. In the LISTOBJECT operation in AWS4 provider files, a new JSON object is added to the requestParameters array. It specifies URL_PARM as the mechanism, and “marker=<GDK_MARKER>” as the description. When CDA is listing objects and needs to start after a certain place, the GDK_MARKER is used. A new JSON object is added to the MESSAGE_BODY mechanism in the responseResults array. The JSON object has the key “partial_list” with a specifics on what to look for when less than the full results are returned for a list request. The content, GDK_IST, indicates the CDA internal variable that is set when a partial list has been returned. When found in the response body, CDA will then issue another request using the GDK_MARKER to indicate where to continue listing from so that the entire results will be returned.

## 1.1.2 New Function Exploitation

In order to use the new functionality, some updates to JCL or an application are required.

### 1.1.2.1 GDKUTIL invocations

- **Create a bucket** - If you use the GDKUTIL utility, you should write JCL that invokes PGM=GDKUTIL with a SYSIN that specifies OPERATION(CREATEBUCKET) to use this operation to create a bucket in the cloud store.
- **Delete a bucke**t - The GDKUTIL SYSIN should specify OPERATION(DELETEBUCKET)
- **Create a bucket with S3 object lock** – The GDKUTIL SYSIN should specify OPERATION(CREATEOBJLOCKBUCKET)
- **List Objects with a common prefix** – The GDKUTIL SYSIN should specify the LIST command with the PREFIX( ) keyword with the wanted object prefix text. e.g. LIST PROVIDER(IBMCOS)

PREFIX(objprefix/lower/) will list all objects that begin with “objprefix/lower/” in the bucket named by the OBJNAME DD.

- **List Objects with names up to a delimiter value**. The GDKUTIL SYSIN should specify the LIST command with the DELIM( ) keyword with the wanted delimiter. e.g. LIST PROVIDER(IBMCOS) DELIM(/) will list all partial object names up to the requested delimiter character.

- **List Buckets with a common prefix** – THE GDKUTIL SYSIN should specify the LIST command with the PREFIX( ) keyword, where the OBJNAME DD only contains the forward slash character (/). This will return a list of all bucket names that begin with the requested prefix text.

- **Ensure all object names are returned for LISTOBJECT requests**. No changes to existing JCL are required. Existing LIST commands on buckets that have more than 1000 objects will now return all object names.

- **Upload multiple UNIX files with one UPLOAD command**. Create new JCL that invokes the GDKUTIL utility with the UPLOAD command in the SYSIN. The OBJNAME DD specifies the bucket name where the objects should be placed. A bucket name starts and ends with the forward-slash character. The PREFIX keyword may be used to indicate a prefix name that every uploaded object should start with. The LOCNAME DD specifies a z/OS UNIX directory name that is the start of the directory tree containing files to be uploaded. The REGEX keyword may be used to specify a regular expression that is used to select which files in the LOCNAME directory tree are selected.

- **Download multiple objects with one DOWNLOAD command**. Create new JCL that invokes the GDKUTIL utility with the DOWNLOAD command in the SYSIN. The OBJNAME DD specifies the bucket name where the objects are downloaded from. A bucket name starts and ends with the forward-slash character. The PREFIX keyword may be used to select only objects beginning with the indicated prefix for processing. The REGEX keyword may be used to further restrict selection of object names to be processed. The LOCNAME DD specifies a UNIX directory where the objects are to be downloaded to. Any pseudo-directory characters (forward-slash) in the object name will result in a UNIX directory with that name being created if one doesn’t already exist.

- **Delete multiple objects with one DELETE command**. Create new JCL that invokes the GDKUTIL utility with the DELETE command in the SYSIN. The OBJNAME DD specifies the bucket name where the objects are deleted from. A bucket name starts and ends with the forward-slash character. The PREFIX keyword may be used to select only objects beginning with the indicated prefix for processing. The REGEX keyword may be used to further restrict selection of object names to be processed.

#### 1.1.2.2 Application usage

In order to use the new functionality from an application, new API calls or optional parameters are required.

- **Create a bucket** – Call the new GDKGEN() API, passing an operation name of CREATEBUCKET.
- **Delete a bucke**t – Call the new GDKGEN() API, passing an operation name of DELETEBUCKET.

- **Create a bucket with S3 object lock** – Call the new GDKGEN() API, passing an operation name of CREATEOBJLOCKBUCKET.

- **List Objects with a common prefix** – When using the GDKLIST() API, passing a bucket name as the object name, along with the “prefix” optional parameter, the returned list GDK_LISTOBJECTS_TYPE buffer will only contain those object names that begin with the prefix value.

- **List Objects with names up to a delimiter value**. When using the GDKLIST() API, passing a bucket name as the object name, along with the “delimiter” optional parameter, the returned list GDK_LISTOBJECTS_TYPE buffer will only contain the partial object names up to the specified delimiter character.

- **List Buckets with a common prefix** –If you use the GDKLIST() API, passing a forward-slash as the object name along with the “prefix” optional parameter, the returned list GDK_LISTOBJECTS_TYPE buffer will only contain the bucket names sharing that prefix.

- **Ensure all object names are returned for LISTOBJECT requests**. When calling the GDKLIST() API, if there are more than 1000 objects in the S3 bucket, the return data will indicate isTruncated. If the provider file has the partial_list object, then CDA will recognize this and use the “marker” functionality to continue to request more names be returned following the previous last name until the GDK_LISTOBJECTS_TYPE buffer is full, or no more object names are returned. No change is needed by the application.

- **Request object names after a specified marker.** When calling the GDKLIST() API, the “marker” optional parameter may be specified. CDA will perform the LIST request requesting the marker support.

- **Get details about an individual object** in a gdklist() call GDK_LISTOBJECTS_TYPE return buffer. Call the GDKLIST() API, passing a full object name.

- **Usage of the GDK_EXIT_DATALOCATION enhancements** – If you have an application that calls the gdkget() and gdkwrite() APIs, and want to exploit the enhancement, then make the following code changes:

    - GDKWRITE() – Ensure that the “Content-Length” and the “Content-LengthE” optional parameters are not being passed in the optionalParm parameter. Optionally specify the “multipartChunkSize” optional parameter with a value greater than 5*1024*1024 to define a buffer size that is filled with data from the Streaming Send Exit.

    - GDKGET() – If you want to retrieve multiple cloud objects in one call, treating them as a single stream of bytes: Pass “regex” as an optional parameter. The value may be the empty string “”, or an actual regular expression that should be used to match object names returned as a result of the LISTOBJECT request with a prefix being the object name. “sort-mechanism” may also be passed as an optional parm with a value of “sort-alpha-with-num”, “sort-alphanum”, or other. The resulting names are sorted as a name that has an alphabetical portion, followed by a numeric part, followed by an optional alphabetical portion when “sort-alpha-with-num” is requested. “sort-alphanum” indicates that the names can be sorted as plain alphanumeric values.

- **Usage of the New APIs**. Read the updates to Chapter 30 in the z/OS MVS Programming: Callable Services for High Level Languages to find the details regarding the new APIs (GDKINIT/GDKTERM, GDKVALD, GDKGEN, and GDKQUERY).

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

SA23-1377-60

Part 11. Cloud Data Access Services is updated as follows:

## 2.1 Chapter 25. Cloud Data Access files

Chapter 25. Cloud Data Access files is updated in the Provider file section as follows:

Before the list of key names and descriptions, replace the description with the following paragraph:

The provider file is a JSON object made up of key names and values. A sample 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. What follows is a list of the recognized key names and their descriptions. The GDKVALD API may be used to retrieve the values associated with key names, allowing applications to retrieve their own application-specific key:value pairs. The following list of key names as well as any prefixed by “cda_” are reserved for CDA usage.

**host**

URI for the cloud object storage provider. May also be a JSON object with a “preferred” key and optional “backup” key that specifies a list of URI values for the cloud object storage provider.

**preferred**

A list of string values with the format: {https://}<uri>{:<port>}{,<sslKeyRingName>} . Values surrounded by curly braces {} are optional. The string value may contain the uri, port number, and sslKey ring name, but only the uri is a required value. When connecting to the host, the host uri entries are attempted to be used, chosen in random order, until one is successfully connected.

**backup**

A list of string values with the same format as the preferred list. The entries in this list are only used when none of the entries in the preferred list can be used.

**cloudserverTZ**

Allows override of the default TZ=GMT as the environment variable for what time zone the current time should be used when calculating the AWS4 signature.

**encodeobjname**

Specifies a string of characters to url-encode when found in the requested object name. For example, z/OS data set names can contain the national characters of #$@. The pound-sign/hash character (#) is a special character for object names, and would need to be url-encoded in order to be used in an object name.

**localIPAddr**

Specifies a value to set on the z/OS Client for Web Enablement Toolkit option, HWTH_OPT_LOCALIPADDR.

**localPort**

Specifies a number value to use as the originating port number in conjunction with the localIPAddr value. Sets the HWTH_OPT_LOCALPORT option.

Under the **authentication** description, add a section as follows:

**credential_schema**

Specifies a schema used to parse the credentials JSON file containing information to build a JSON Web Token (JWT)

**JWT_Duration**

Specifies the length of time to request validity of the JWT in minutes.

**JWT_claims**

A JSON object containing the claims used to build the JWT.

The **supportedOperations** description is updated to add the following names to the name list: LISTBUCKETS, HEADOBJECT, CREATEBUCKET, and DELETEBUCKET. Add another sentence after the list as follows:

Additional operations may be added to the supportedOperations array with unique names to be used with the GDKGEN API.

The **requestParameters** description is updated to add a new entry under the **mechanism** list:

**URL_PARM**

Add url parameters to the url before signing and issuance of the HTTP command. The url will have the format of: <url>?<descriptor>&<descriptor>

Three new bullets are added under the **descriptor**:

- <GDK_PREFIX>

  The value passed on the “prefix” optional parameter.
- <GDK_DELIMITER>

  The value passed on the “delimiter” optional parameter.
- <GDK_MARKER>

  The value passed on the “marker” optional parameter.
- <GDK_VERSION_ID>

  The value passed on the “versionID” optional parameter.

**responseResults**

Allows definition of how to parse the response from the request.

**mechanism**

HEADER to indicate this is a rule to be used when processing response headers, or MESSAGE_BODY to indicate this is the rule to bue used when processing the response body.

**name**

Used in a mechanism: HEADER rule. The value is matched with the HTTP header name.

**content**

The value specifies a CDA variable that the value from the header is stored under for later retrieval.

**contentType**

Specifies the type of content being returned from the object server. Common options are: application/xml, application/json, text/plain

**schema**

A JSON object that describes a schema used to interpret the content of the response body.

**b64_urlEncodeGDKPART**

"true" indicates that the multipart part numbers need to be base64 url encoded.

**elementName**

The XML tag or JSON key name to match.

**xmlns**

The XML namespace value to add to the XML document being built as the request body.

**entries**

Descriptor of an array of XML entries.

**type**

Indicator of the type of value. "object", "string", "array", and "number" are possible values.

**item**

Describes one entry in the "entries" array.

**modifiedTimeFormatP**

Describes how to parse a returned object modified time value according to the strftime() XLC runtime function.

**createdTimeFormatP**

Describes how to parse a returned object created time value according to the strftime() XLC runtime function.

## 2.2 Chapter 26. Cloud Data Access (CDA) API basics

Chapter 26. Cloud Data Access (CDA) API basics – Elements of Cloud Data Access is updated to add the following new bullets:

- Start a CDA Session that will contain 0 or more related API calls.
- Stop a CDA Session and perform cleanup of the session.
- Perform a named operation from the provider file.
- Validate a provider file and retrieve values from that provider file.
- Query availability of CDA functions.

## 2.3 Chapter 29. Syntax, linkage, and programming considerations

Chapter 29. is updated to add entries in the Direct Linkage table as follows:

| API Entry Offset from DFVCDAVT | API Name | Language Environment Requirement |
|---|---|---|
| 52 (34) | GDKGEN | YES |
| 56 (38) | GDKGENN | NO |
| 60 (3C) | GDKINIT | YES |
| 64 (40) | * | * |
| 68 (44) | GDKTERM | YES |
| 72 (48) | * | * |
| 76 (4C) | GDKVALD | YES |
| 80 (50) | GDKVALDN | NO |
| 84 (54) | GDKQUERY | YES |
| 88 (58) | GDKQRYN | NO |
| 92 (5C) | GDKMSGTL | YES |

## 2.4 Chapter 30. Cloud Data Access callable services

Chapter 30. Cloud Data Access callable services is updates as follows:

The bullet list that lists the callable services and their page number is updated to add the following lines:

- “GDKINIT – Initialize a CDA Session”
- “GDKTERM – Terminate a CDA Session”
- “GDKGEN – Perform a configurable request”
- “GDKVALD – Validate a provider file”
- “GDKQUERY – Query CDA functionality”

### 2.4.1 GDKGET – Retrieve a Cloud Object

The **dataLocationPtr** section is updated to enhance the **GDK_EXIT_DATALOCATION** description to add a new paragraph as follows:

The GETLARGEOBJECT operation can be utilized by the caller’s streaming receive exit to break up a large object across multiple HTTP requests. The “regex” optional parameter should be used to indicate that the GETLARGEOBJECT operation is preferred. The “regex” functionality can also be used to stream multiple objects yet be seen by the caller’s streaming exit as a single object.

The accepted optional parameter table is updated to add the following row:

| NAME | Type | Description |
|---|---|---|
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |
| “regex” | Address | Specifies the address of a null-terminated string value that is to be used as a regular expression determining which objects are streamed via the caller’s streaming receive exit. (Only valid when GDK_EXIT_DATALOCATION is specified as the dataLocationType.)<br>When specified with a non-empty value, the objectName value is used as a PREFIX to retrieve a list of objects that begin with that name. The GETLARGEOBJECT operation |

| | | |
|---|---|---|
| | | is used to stream each of the objects to the caller's streaming exit.<br>When specified with an empty value, the objectName is retrieved using the GETLARGEOBJECT operation and streamed to the caller's streaming exit. |
| "sort-mechanism" | Address | Specifies the address of a null-terminated string value indicating the sorting mechanism used to order the list of objects that are retrieved. (Only examined when "regex" is a non-empty value.) Acceptable values are:<br>• "sort-alpha-with-num" – The results are sorted on the alphabetical portion followed by the number value. If you know that the object names returned from a LIST with PREFIX have the form <text><num><text>, this sorting mechanism should be used.<br>• "sort-alphanum" – The names matched can be sorted in alphanumeric order. i.e. Name1 compared to Name2.<br>If not specified, then the order of object names returned from the cloud server is how they are delivered to the caller's streaming receive exit. |
| "max_objs" | Address | Specifies the address of a null-terminated string value indicating the storage to contain the maximum amount of object names when finding the matching object names. If not specified, the default value is 1000. |
| "prefix" | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_PREFIX substitution text in a URL_PARM request parameter. |
| "delimiter" | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_DELIMITER substitution text in a URL_PARM request parameter. |
| "versionID" | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_VERSION_ID substitution text in a URL_PARM request parameter. |

A new return code is added to the GDKGET return code values table as follows:

| | | |
|---|---|---|
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified <cloudProvider>.json file is empty. <cloudProvider> is the provider name specified on the API parameter list. |

### 2.4.2 GDKWRITE – Write an object to cloud storage

The **dataLocationPtr** section is updated to enhance the **GDK_EXIT_DATALOCATION** description to add a new paragraph as follows:

The REST API for some cloud storage providers require that a "Content-Length" header is included on every request. If you know the total amount of data that will be sent via the streaming exit, then you should use the "Content-Length" or "Content-LengthE" optional parameters to pass that value to be included on the request. After application of OA65224, support is added to buffer data from the streaming exit so that the Multipart Upload support described by WRITELARGEOBJECT can be used. When no content length is specified via the optional parameters, the Multipart Upload support will be used.

The size of the buffer may be modified by the “multipartChunksize” optional parameter. If not specified, the buffer is 8MB in size. If the caller’s streaming send exit sets the state to HWTH_STREAM_SEND_EOD before the first buffer has reached 5MB, the data will be sent in a single request, not utilizing the Multipart Upload feature.

The **optionalParmStructPtr** section is updated as follows:

The accepted optional parameter table is updated to add the following row:

| **NAME** | **Type** | **Description** |
|---|---|---|
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |
| "multipartChunksize” | Unsigned number | Specifies the size of a single chunk of data contained in a Multipart Upload request (described by the WRITELARGEOBJECT operation). The minimum value is 5242880. |
| “prefix” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_PREFIX substitution text in a URL_PARM request parameter. |
| “delimiter” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_DELIMITER substitution text in a URL_PARM request parameter. |
| “versionID” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_VERSION_ID substitution text in a URL_PARM request parameter. |

A new return code is added to the GDKGET return code values table as follows:

| | | |
|---|---|---|
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified <cloudProvider>.json file is empty. <cloudProvider> is the provider name specified on the API parameter list. |

### 2.4.3 GDKDEL – Delete an object from cloud storage

The accepted optional parameter table is updated to add the following row:

| **NAME** | **Type** | **Description** |
|---|---|---|
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |
| “prefix” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_PREFIX substitution text in a URL_PARM request parameter. |
| “delimiter” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_DELIMITER substitution text in a URL_PARM request parameter. |

| | | |
|---|---|---|
| “versionID” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_VERSION_ID substitution text in a URL_PARM request parameter. |

A new return code is added to the GDKGET return code values table as follows:

| | | |
|---|---|---|
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified <cloudProvider>.json file is empty. <cloudProvider> is the provider name specified on the API parameter list. |

### 2.4.4 GDKLIST – List cloud objects

The **Description** is to be updated to the following:

The GDKLIST API is used to retrieve a list of buckets, a list of objects in a bucket, or a single object for a cloud provider. When called with a cloud provider and a bucket name and a buffer, an HTTP request is sent to the cloud object storage server to retrieve the information according to the definition in the LISTBUCKETS, LISTOBJECT, and HEADOBJECT entries in the supportedOperations array in the cloud provider json file, is sent and information about the cloud objects returned.

The GDKLIST API processing can additionally be modified through the optionalParmStruct.

The **Parameters** section is updated to modify the bucketName description. The bucketName description first paragraph is replaced by:

Specifies the address of a pointer to a name that should be listed. The name can be one of the following:

- / - A single forward slash indicates that all bucket names should be returned in the list. This list may be modified by the “filter” optional parameter. (The LISTBUCKETS operation must exist in the provider file.)
- /<bucket_name>/ - A list of all objects within that bucket_name should be returned along with the object size and creation date. This list may be modified by the “prefix” and “delimiter” optional parms. (The requestParameter object for the URL_PARM mechanism, either for GDK_PREFIX or GDK_DELIMITER, must exist in the LISTOBJECT operation.)
- /<bucket_name>/<object_name> - A specific object should be listed and details returned. (The HEADOBJECT operation must exist in the provider file

The name passed must be null-terminated. The bucket/container name is folded to lowercase. The name is not checked for valid URL characters. For more information, see RFC 3986 (www.ietf.org/rfc/rfc3986.txt).

The **optionalParmStructPtr** description is updated. The optional parameters table is updated to add the following rows:

| NAME | Type | Description |
|---|---|---|
| “prefix” | Address | Specifies the address of a null-terminated string that is the common object prefix that should be returned. This value would be placed on the PREFIX parameter of the request url. This is only done if the requestParameters object for the LISTOBJECTS operation has a “URL_PARM” mechanism. When prefix is specified, but a URL_PARM for prefix is not found, an error will be returned. |

| | | |
|---|---|---|
| “delimiter” | Address | Specifies the address of a to a single char to be used on the delimiter parameter of the request url. This value would be placed on the DELIMiter parameter of the request url. This is only done if the requestParameters object for the LISTOBJECTS operation has a “URL_PARM” mechanism. When delimiter is specified, but a URL_PARM for delimiter is not found, and error will be returned. |
| “filter” | Address | Specifies the address of a null-terminated string that should be used to compare with the beginning of each bucket name returned. If the entire filter matches the beginning of the bucket name, the name is included in the returned results. |
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |
| “marker” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_MARKER substitution text in a URL_PARM request parameter. |
| “versionID” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_VERSION_ID substitution text in a URL_PARM request parameter. |

A new return code is added to the GDKGET return code values table as follows:

| | | |
|---|---|---|
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified <cloudProvider>.json file is empty. <cloudProvider> is the provider name specified on the API parameter list. |

### 2.4.5 GDKMSGTR – Translate a CDA return code into text

The accepted optional parameter table is updated to add the following row:

| NAME | Type | Description |
|---|---|---|
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |

### 2.4.6 GDKGETP – List cloud providers

The accepted optional parameter table is updated to add the following row:

| NAME | Type | Description |
|---|---|---|
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |

A new return code is added to the GDKGET return code values table as follows:

| | | |
|---|---|---|
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified <cloudProvider>.json file is empty. <cloudProvider> is the provider name specified on the API parameter list. |

### 2.4.6.1 GDKINIT – Initialize a Session

A new section, titled "GDKINIT – Initialize a CDA Session", is added after the GDKKEYGR section. Contents are as follows:

The GDKINIT API allows callers to start a 'CDA session' to be used with the other CDA APIs. A session can cross multiple API calls. Once a session has been started, some things will be skipped for subsequent API calls. Those are:

- The config.json file is not re-read.
- The provider file for the session is not re-read.
- The key file for the session is not re-read.
- Once a REST API call has been made, the SSL connection is kept open to be used for subsequent REST API calls to the same cloud.

The GDKTERM API must be called to terminate and cleanup the session once complete.

Even though the key file is not re-read, the ICSF APIs are still used to decrypt the cloud credentials when they are needed to ensure that the cloud credentials are not kept in the clear longer than necessary to perform the requested action.

The GDKINIT API processing can additionally be modified through the optionalParmStruct.

**Purpose (or Function):** Initialize a Cloud Data Access session that will consist of 0 or more DFSMSdfp CDA API calls.

**Requirements:** When using the Branch Entry, a Language Environment must be established before the gdkinit() API is invoked. When called from a Language Environment C program, the environment is already created and available. Otherwise, a PreInitialization environment can be used to create a Language Environment (CEEPIPI). The z/OS Language Environment Programming Guide has more details in the "Using preinitialization services" section.

When linking with SYS1.CSSLIB(GDKCSS), then a Language Environment should already be established. There is no entry in the SYS1.CSSLIB(GDKCSSNL) for GDKINIT.

**Format:**

```
gdkinit (retCodeAddr,
         cloudProvider,
         session_handle,
        optionalParmStructPtr);
```

**Parameters:**

parameter name description

**retCodeAddr**

Specifies the address of a 4-byte field that the API will place the return code into.

**cloudProvider**

Specifies the address of a pointer to a name of the cloud provider that all requests for this session will use. The name must be null-terminated. When retrieving the CloudProvider definition corresponding to the cloudProvider specified on the API call, the user's RACF ID, or UserID from the optionalParms, is used to examine the associated OMVS segment to retrieve the home directory. That home directory is examined for a gdk/providers/ sub-directory. If it exists and the <cloudProvider>.json file is found, it will be used as the template to communicate with the cloud provider. If the gdk/providers/ directory does not exist or the <cloudProvider>.json file is not found there, then the CDA System Default directory of /usr/lpp/dfsms/gdk/providers/ will be used.

Subsequent calls to the CDA APIs that specify a Cloud Provider name that communicate with the Cloud Server will ignore the specified name and instead use the Cloud Provider for the session. The Cloud Provider JSON file is only read once per session.

The application may request a list of all supported providers by using the **GETPROVIDERS** operation.

**session_handle**

Specifies the address of a 12 byte field where DFSMSdfp CDA will store a session handle that uniquely identifies the session.

**optionalParmStructPtr**

Specifies an optional method for a user of this API to provide customized processing not provided by default by the CDA API. The API will specify a pointer to a structure as mapped by the data structure **GDK_OPTIONAL_PARMS_TYPE**. This data structure will contain one or more customized overrides or additions. In general, all string values, including keys and values, must be null terminated when being passed to the APIs.

| Optional Parms | | |
|---|---|---|
| **NAME** | **Type** | **Description** |
| "UserID" | 8-byte char | RACF User ID used to retrieve Cloud security credentials for the session. |
| "AutoConvert" | Character | "true" means that ASCII to EBCDIC translation should be performed on the data being sent or retrieved from the cloud<br>"false" means that no translation should be performed.<br>This optional parameter will override any default, or value in the config.json file. If this value is passed, this sets the new default for the session. The default may be overridden for individual subsequent API calls, but the override only lasts for that call.<br>If not passed, the default for the session is no translation of data. |

|  |  |  |
|---|---|---|
| “Use-Config-File” | Character | “false” means that the config.json file should not be read for default configuration values.<br>Any other values mean that the config.json should be used.<br>On subsequent API calls, the “Use-Config-File” parm is ignored because the default config has been set by the gdkinit() call. |
| “log-level” | Character | This setting sets the default logging level for all subsequent API calls. The default logging level may be overridden for specific API calls via the optional Parms on that call. The default 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. The levels in order of low to high severity are listed:<br>“DEBUG” means all logging messages are written to.<br>“INFO” means only INFO and higher severity logging messages are written.<br>“NOTICE” means only NOTICE and higher severity logging messages are written.<br>“WARNING” means only WARNING and higher severity logging messages are written.<br>“ERROR” means only ERROR logging messages are written.<br>“NONE” means no messages are written regardless of severity. |
| "web-toolkit-logging" | Character | This parameter sets the default value of the web-toolkit-logging for the session. The value for a specific API call may be overridden by the optional Parms of that API call.<br>“false” means that additional logging messages from the z/OS Client Web Enablement Toolkit processing should not be written.<br>“true” means that additional logging messages from the z/OS Client Web Enablement Toolkit processing should be written.<br>This value will override any default or value in the config.json file. Logging messages are written to stdout. |

**Usage (or Usage Notes):**

When the API is invoked, if UserID wasn’t provided in the OptionalParms, the current user’s RACF ID is used to retrieve the applicable CloudProvider definitions, as well as read the gdkkeyf.json document for decryption by subsequent API calls. The gdkkeyf.json document will not be re-read for any subsequent API calls that interact with the Cloud Storage server. It may be re-read as needed for subsequent Key Management API calls.

When retrieving the CloudProvider definition corresponding to the cloudProvider specified on the API call, the user’s RACF ID, or UserID from the optionalParms, is used to examine the associated OMVS segment in order to retrieve the home directory. That home directory is examined for a gdk/providers/ sub-directory. If it exists and the <cloudProvider>.json file is found, it will be used as the template to communicate with the cloud provider. If the gdk/providers/ directory does not exist or the provider file is not found there, then the CDA System Default directory of /usr/lpp/dfsms/gdk/providers/ will be used.

When retrieving the Security Credentials to use to communicate with the cloudProvider specified on the API call, the gdkkeyf.json file must first be located. The user’s RACF ID, or UserID from the optionalParms, is used to find the home directory. That home directory is examined for a gdk/gdkkeyf.json document that can be accessed for READ. If not found, then the CDA API call fails.

**Restrictions:** None.

**Authorization (or Privilege required):** User

**Related Information (or Related Services):** GDKGET, GDKWRITE, GDKDEL, GDKLIST

**Return Value (or Return and reason code):**

The various return code constants are documented in the gdkic header file, found in SYS1.SIEAHDRV.H.

| Code | Constant Name | Explanation |
|---|---|---|
| 0 | GDK_OK | Processing was successful |
| 100 | GDK_UNABLE_TO_READ_KEYFILE | The gdkkeyf.json document was not found in the CDA System directory |
| 104 | GDK_CLOUD_PROVIDER_NOT_FOUND | When reading the gdkkeyf.json document, for the requested user, the specified cloudProvider was not found. |
| 110 | GDK_PROVIDER_OPEN_FAILURE | The JSON document for the requested cloudProvider does not exist, or was found, but cannot be opened for READ. |
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified <cloudProvider>.json file is empty. <cloudProvider> is the provider name specified on the API parameter list. |
| 799 | GDK_UNEXPECTED_ERROR | An unexpected error occurred. Contact IBM L2, and provide the logging output. |

**Examples:** TBD.

**Other:** None.

**Notes to Information Solutions:** TBD

#### 2.4.6.2 GDKTERM – Terminate a Session

A new section, titled “GDKTERM – Terminate a CDA Session”, is added after the GDKINIT section. Contents are as follows:

The GDKTERM API is called to indicate that a ‘CDA session’ is complete, and any cleanup should be performed. Any open connection is closed and cleaned up. Any storage obtained across the session is released.

**Purpose (or Function):** Terminate a Cloud Data Access session that was established via a GDKINIT API call.

**Requirements:** When using the Branch Entry method, a Language Environment must be established before the gdkterm() API is invoked. When called from a Language Environment C program, the environment is already created and available. Otherwise, a PreInitialization environment can be used to create a Language Environment (CEEPIPI). The z/OS Language Environment Programming Guide has more details in the “Using preinitialization services” section.

When linking with SYS1.CSSLIB(GDKCSS), then a Language Environment should already be established. There is no entry in the SYS1.CSSLIB(GDKCSSNL) for GDKTERM.

**Format:**

```
gdkterm (retCodeAddr,
         session_handle,
        optionalParmStructPtr);
```

**Parameters:**

parameter name    description

**retCodeAddr**

Specifies the address of a 4-byte field that the API will place the return code into.

**session_handle**

Specifies the address of a 12 byte field where DFSMSdfp CDA will store a session handle that uniquely identifies the session.

**optionalParmStructPtr**

Specifies an optional method for a user of this API to provide customized processing not provided by default by the CDA API. The API will specify a pointer to a structure as mapped by the data structure **GDK_OPTIONAL_PARMS_TYPE**. This data structure will contain one or more customized overrides or additions. In general, all string values, including keys and values, must be null terminated when being passed to the APIs.

| Optional Parms | | |
|---|---|---|
| **NAME** | **Type** | **Description** |
| “log-level” | Character | This setting overides the default logging level for this API call. This value will override any default, or value in the config.json file. Logging messages are written to stdout. The levels in order of low to high severity are listed:<br>“DEBUG” means all logging messages are written to.<br>“INFO” means only INFO and higher severity logging messages are written.<br>“NOTICE” means only NOTICE and higher severity logging messages are written.<br>“WARNING” means only WARNING and higher severity logging messages are written.<br>“ERROR” means only ERROR logging messages are written. |

| | | |
|---|---|---|
| | | “NONE” means no messages are written regardless of severity. |

**Usage (or Usage Notes):**

TBD

**Restrictions:** None.

**Authorization (or Privilege required):** User

**Related Information (or Related Services):** GDKWRITE, GDKDEL, GDKLIST

**Return Value (or Return and reason code):**

The various return code constants are documented in the gdkic header file, found in SYS1.SIEAHDRV.H.

| Code | Constant Name | Explanation |
|---|---|---|
| 0 | GDK_OK | Processing was successful |
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 799 | GDK_UNEXPECTED_ERROR | An unexpected error occurred. Contact IBM L2, and provide the logging output. |

**Examples:** TBD.

**Other:** None.

**Notes to Information Solutions:** TBD

### 2.4.6.3 GDKGEN – Perform a Configurable request

A new section, titled “GDKGEN – Perform a Configurable request”, is added after the GDKTERM section. Contents are as follows:

The GDKGEN API will allow callers to request that CDA perform an operation found in the supportedOperations array in the specified provider file. Having a generic (configurable) API will allow CDA to perform other operations such as CreateBucket, DeleteBucket, GetObjectACL, etc.

When called, the provider definition for the requested cloud provider will be retrieved and the security credentials for the invoking user will be retrieved. Those security credentials will be used to send an HTTP request as described by the named Operation.

The GDKGEN API processing can additionally be modified through the optionalParmStruct.

**Purpose (or Function):** Perform a named operation found in the provider file supportedOperations array. The API allows for invocation of a configurable request, where the programmer knows the name of the operation in the provider file, as well as what inputs and outputs are expected for that particular operation.

**Requirements:** When using the Branch Entry, a Language Environment must be established before this API is invoked. When called from a Language Environment C program, the environment is already created and available. Otherwise, a PreInitialization environment can be used to create a Language Environment (CEEPIPI). The z/OS Language Environment Programming Guide has more details in the “Using preinitialization services” section.

When linking with SYS1.CSSLIB(GDKCSS), then a Language Environment should already be established. When linking with SYS1.CSSLIB(GDKCSSNL), then a Language Environment does not need to be established as it will be created on entry, and terminated at exit.

**Format:**

```
gdkgen (retCodeAddr,
        cloudProvider,
        objectName,
        operation,
        genParmStructPtr,
        optionalParmStructPtr);
```

When using Branch Entry, and a Language Environment is not established, the GDKGENN API may be used. This will create a Language Environment upon entry and terminate the Language Environment upon exit.

**Format:**

```
gdkgenn (retCodeAddr,
         cloudProvider,
         objectName,
         operation,
         genParmStructPtr,
         optionalParmStructPtr);
```

**Parameters:**

parameter name description

**retCodeAddr**

Specifies the address of a 4-byte field that the API will place the return code into.

**cloudProvider**

Specifies the address of a pointer to a name of the cloud provider that this request should contact. The name must be null-terminated. When retrieving the CloudProvider definition corresponding to the cloudProvider specified on the API call, the user’s RACF ID, or UserID from the optionalParms, is used to examine the associated OMVS segment to retrieve the home directory. That home directory is examined for a gdk/providers/ sub-directory. If it exists and the <cloudProvider>.json file is found, it will be used as the template to communicate with the cloud provider. If the gdk/providers/ directory does not exist or the <cloudProvider>.json file is not found there, then the CDA System Default directory of /usr/lpp/dfsms/gdk/providers/ will be used.

The application may request a list of all supported providers by using the **GETPROVIDERS** API.

**objectName**

Specifies the address of a pointer to the remote object name to be used during processing. The remote object name is referenced as a whole as GDK_OBJECT in the provider file. It is further broken down into to variables referenced as GDK_OBJECT_PART and GDK_BUCKET_PART, where GDK_BUCKET_PART is the characters between the first forward slash, and the second forward slash. This value must specify the complete remote cloud provider location description including bucket/container and/or path and/or file name that is the target of this request (e.g. /bucket2/dir1/CDA.pdf). The name must start with a forward-slash (/), and the bucket/container name is the characters up to the next forward-slash (/). The name passed must be null-terminated. The bucket/container name is folded to lowercase. The name is not checked in general for valid URL characters. see: www.ietf.org/rfc/rfc3986.txt When interacting with a provider using the AWS4 authentication model, some characters are encoded as follows:

| Character | Encoded version |
|---|---|
| / | %2F |
| "encode": "special" in provider file | |
| ! | %21 |
| * | %2A |
| ( | %28 |
| "encodeUrlChars" in provider file will cause the specified characters to be URL encoded | |

**operation**

Specifies the address of a pointer to the name of the matching operation from the supportedOperations array in the provider file. The name is folded to uppercase before attempting to match with an operation in the supportedOperations array. The name must be null-terminated.

**genParmStructPtr**

Specifies additional information or data that is intended to be used by DFSMSdfp CDA during the processing of the requested operation. The operation as defined in the provider file may require some data as identified by the <var_name> syntax in the provider file. e.g. <GDK_DATA> identifies a pointer to a buffer of data to be sent to the Cloud Server.

In a single GDK_GEN_TYPE entry, the fields are as follows:

- GDK_genKey is a pointer to a null terminated string of the variable name that this entry describes
- GDK_genValue is a pointer to storage that is defined via the GDK_genType and GDK_genLen values.
- GDK_genType is a 2 byte unsigned number that indicates what GDK_genValue points to. I.e. a 4 byte integer, or a character array of EBCDIC data.
- GDK_genLen is a 2-byte unsigned number that indicates the length of the storage that GDK_genValue points to. Before calling the GDKGEN() API, this field should be initialized to the amount of storage available. After the GDKGEN() API call returns, it will contain the amount of storage used for the return value. For example: A GDK_genLen of 4 paired with GDK_genType of GDK_GEN_INT means that GDK_genValue points to a 4-byte integer value.

genericParmsStruct Control Block Mapping

```
/* ---------------------------------------------------------------- */
/* genericParmsStruct Type definition (GDKGEN service)              */
/*   - maps the data pointed to by the genParmStructPtr             */
/*     parameter                                                    */
/* ---------------------------------------------------------------- */
typedef void * GDK_GEN_VALUE_TYPE; /* Generic pointer to value      */

typedef struct {
  GDK_OPTION_KEY_TYPE GDK_genKey;  /* Key name pointer              */
  GDK_GEN_VALUE_TYPE  GDK_genValue; /* Value pointer                */
  unsigned short      GDK_genType;  /* Type of value                */
  unsigned short      GDK_genLen;   /* Length of value area         */
} GDK_GEN_TYPE;                    /* Single Generic Entry          */

typedef struct {
  uint32_t GDK_NumOfGenTypes;      /* Count of Generic Entries      */
  GDK_GEN_TYPE GDK_GenArray[];     /* Array of GDK_GEN_TYPEs        */
} GDK_GEN_PARMS_TYPE;              /* Header of structure           */

#define GDK_GEN_STRING 0;          /* String type (default)         */
#define GDK_GEN_INT    1;          /* Integer type                  */
#define GDK_GEN_ADDR   2;          /* Address type                  */
#define GDK_GEN_FLOAT  3;          /* Floating point type           */
```

The name of the GDK_Gen_Key may be selected as preferred. When following the rules defined in the provider file, the key name will be used. There are some CDA internal variables that are used internally, and if specified, may not be honored if passed via the genericParmsStruct.

The following table describes CDA variables that may be specified by the GDK_GEN_TYPE entry:

| CDA Variable Name | Description |
|---|---|
| GDK_DATA_LOCATION | Address of the thing where the data is read from, or written to. Requires GDK_DATA_LOCATION_TYPE to also be passed to describe what is pointed to. |
| GDK_DATA_LOCATION_TYPE | Integer. Describes the thing that GDK_DATA_LOCATION points to.<br><br>1 – Buffer area. GDK_DATA_LOCATION is a 31-bit address of an area to read data from, or write data to. GDK_DATA_LOCATION_LEN is the length of data in the buffer when reading, or the maximum size of the buffer.<br><br>2 – UNIX file Path or Data Set name. GDK_DATA_LOCATION is a 31-bit address that points to the name of a z/OS UNIX data set, or data set name that contains the data to read, or should be written to with data received. GDK_DATA_LOCATION_LEN is the length of the null-terminated path name. Name must conform to conditions described in the z/OS XL C/C++ Programming Guide, Input and Output chapter, performing OS I/O operations -> Opening Files → Using fopen() or freopen(). |

| | |
|---|---|
| | • UNIX file absolute path. Begins with /, and includes all directories between root and the file name. I.e. /u/user1/file.txt<br>• Data Set name. Begins with //'DATASET.NAME'<br><br>3 – Streaming Exit pointer. GDK_DATA_LOCATION is a 31-bit address that points to executable code that conforms to the z/OS Client for Web Enablement Toolkit streaming exit expectations. GDK_DATA_LOCATION_LEN must be the value 4, indicating it is a 31-bit address |
| GDK_DATA_LOCATION_LEN | 4 byte Unsigned integer. Describes the length of the data that GDK_DATA_LOCATION points to. |
| GDK_READ_DATA_LEN_PTR | Address of a 4-byte unsigned integer. On an operation that has read data from the Cloud Object Store, the amount of data read will be placed in this integer. |

The following table describes existing CDA internal variable names and their descriptions. These variables will be overridden by CDA internal processing.

| CDA Variable Name | Description |
|---|---|
| GDK_DATA | Pointer to data buffer used in operation |
| GDK_DATA_LEN | Length of data in buffer for WRITE type operations |
| DATE_ISO_8601 | GMT current timestamp in ISO 8601 format |
| AZURE_ACCOUNT | Account name from saved Cloud Credentials |
| DATE_GMT | GMT current timestamp |
| GDK_OBJECT_NAME | The remote object name is referenced as a whole. (Bucket name and object name.) |
| GDK_PART | Part number from a multipart upload |
| GDK_GMT_UNIX_TIME | GMT current timestamp in number of seconds since January 1, 1970 |
| GDK_JWT_EXP | JWT Expiration time calculated from the GDK_GMT_UNIX_TIME + JWT_ JWT_duration value from provider file. |
| HOST | Host value from Provider file |
| PARAMETER_SET | Used internally by DFSMSdfp CDA for collecting requestParameters from the provider file. |
| UPLOADID | Used internally by DFSMSdfp CDA for tracking the Upload ID during a Multi-Part upload |

| | |
|---|---|
| GDK_ETAG | Used internally by DFSMSdfp CDA for tracking the returned eTag during a Multi-Part upload. |
| GDK_PREFIX | Set from the “prefix” optional parm, and used in URL_PARM request parameters. |
| GDK_DELIMITER | Set from the “delimiter” optional parm, and used in URL_PARM request parameters. |
| GDK_MARKER | Set from the “marker” optional parm, or set internally to the last object in a list when GDK_IST is set, and used in URL_PARM request parameters. |
| GDK_IST | Indicates that a truncated list of objects was found. |
| GDK_VERSION_ID | Set from the “prefix” optional parm, and used in URL_PARM request parameters. |

**optionalParmStructPtr**

Specifies an optional method for a user of this API to provide customized processing not provided by default by the CDA API. The API will specify a pointer to a structure as mapped by the data structure **GDK_OPTIONAL_PARMS_TYPE**. This data structure will contain one or more customized overrides or additions. In general, all string values, including keys and values, must be null terminated when being passed to the APIs.

| Optional Parms | | |
|---|---|---|
| **NAME** | **Type** | **Description** |
| “UserID” | 8-byte char | RACF User ID used to retrieve Cloud security credentials for the GET request. |
| “Content-Length” | Integer | Integer value to place in the HTTP header of the request indicating the amount sent. |
| “Content-LengthE” | 8-byte Integer | Address of an 8-byte Integer value to place in the HTTP header of the request indicating amount sent. |
| “Range-Start” | Integer | Specifies the byte number for the data retrieve to start at. Values are 0 based and inclusive. If not specified, then it is assumed to start at 0. If invalid, may be ignored by Cloud Server. |
| “Range-End” | Integer | Specifies the ending byte number for the request. If not specified (and Range-Start is specified), then read to end. If Range-End is specified (or is 0), but Range-Start is not specified, then the last number of bytes, inclusive, are returned from the object. If invalid, may be ignored by Cloud Server. |
| “Get-HWTH-Code” | Address | Specifies the address of a 4-byte Integer field where the resulting z/OS Client Web Enablement Toolkit return code should be stored. (See HWTRQST documentation in the z/OS Client Web Enablement Toolkit HTTP Enabler section of the z/OS MVS Programming: Callable Services for High Level Languages manual for more details.) |
| “Get-HWTH-Diag” | Address | Specifies the address of an HWTH_DIAGAREA_TYPE area where the resulting z/OS Client Web Enablement Toolkit diagnostic information should be stored. (See HWTRQST documentation for more details.) |

| | | |
|---|---|---|
| “AutoConvert” | Character | “true” means that ASCII to EBCDIC translation should be performed on the data being retrieved from or sent to the cloud.<br>“false” means that no translation should be performed.<br>This optional parameter will override any default, session value, or value in the config.json file.<br>If not passed, the default is no translation of data, or whatever the session value is. |
| “Set-Header-Buffer” | Character | String of custom headers that should be included on the HTTP GET request. Each header is newline (\n) separated. |
| “Get-Header-Buffer” | Address | Pointer to storage area to place the headers from the HTTP Response. |
| "Get-Header-Buffer-Size" | Integer | Size of the provided header buffer. Must be provided if “Get-Header-Buffer” is specified. |
| “Get-Body-Buffer" | Address | Pointer to storage area to place the return body from the HTTP Response. |
| “Get-Body-Buffer-Size" | Integer | Size of the provided Body buffer. Must be provided if “Get-Body-Buffer" is specified. |
| “Get-Response-Code” | Address | Pointer to 4-byte area to place the HTTP status code from the HTTP GET request. |
| “Use-Config-File” | Character | “false” means that the config.json file should not be read for default configuration values.<br>Any other values mean that the config.json should be used. |
| “log-level” | Character | 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 stdout. The levels in order of low to high severity are listed:<br>“DEBUG” means all logging messages are written to.<br>“INFO” means only INFO and higher severity logging messages are written.<br>“NOTICE” means only NOTICE and higher severity logging messages are written.<br>“WARNING” means only WARNING and higher severity logging messages are written.<br>“ERROR” means only ERROR logging messages are written.<br>“NONE” means no messages are written regardless of severity. |
| "web-toolkit-logging" | Character | “false” means that additional logging messages from the z/OS Client Web Enablement Toolkit processing should not be written.<br>“true” means that additional logging messages from the z/OS Client Web Enablement Toolkit processing should be written.<br>This value will override any default or value in the config.json file. Logging messages are written to stdout. |
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |

|  |  |  |
|---|---|---|
| “prefix” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_PREFIX substitution text in a URL_PARM request parameter. |
| “delimiter” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_DELIMITER substitution text in a URL_PARM request parameter. |
| “versionID” | Address | Specifies the address of a null-terminated string value that should be stored in the GDK_VERSION_ID substitution text in a URL_PARM request parameter. |

**Usage (or Usage Notes):**

When the API is invoked without the session handle, if UserID wasn’t provided in the OptionalParms, the current user’s RACF ID is used to retrieve the applicable CloudProvider definitions, as well as retrieve the appropriate Security Credentials for the User/Cloud Provider/Resource combination from the gdkkeyf.json document.

When retrieving the CloudProvider definition corresponding to the cloudProvider specified on the API call, the user’s RACF ID, or UserID from the optionalParms, is used to examine the associated OMVS segment in order to retrieve the home directory. That home directory is examined for a gdk/providers/ sub-directory. If it exists and the <cloudProvider>.json file is found, it will be used as the template to communicate with the cloud provider. If the gdk/providers/ directory does not exist or the provider file is not found there, then the CDA System Default directory of /usr/lpp/dfsms/gdk/providers/ will be used.

When retrieving the Security Credentials to use to communicate with the cloudProvider specified on the API call, the gdkkeyf.json file must first be located. The user’s RACF ID, or UserID from the optionalParms, is used to find the home directory. That home directory is examined for a gdk/gdkkeyf.json document that can be accessed for READ. If not found, then the CDA API call fails.

Once the keyfile has been found, it is parsed, looking for an entry associated with the user’s RACF ID, or UserID from the optionalParms. The most specific bucket/container name (Resource) is searched for first. A bucket name is the first characters surrounded by a forward slash character. i.e. /bucket_name/ If no entry is found, then the generic / Resource name is searched for.

With the most appropriate entry found, the Security Credentials will be decrypted using ICSF. The decryption key is expected to be found using a keylabel of GDK.<cloudProvider>.<UserID/RACF ID>.<sequence_number> . If RACF protection of the ICSF key labels is being used, the invoker of the API must have authority to the key label.

If a valid session handle is passed via the optional parameters, then the config.json file, keyfile, and provider file are not re-read.

Ensure that the GDK variables that are specified in the provider file for the requested operation are completely satisfied. GDK variables are in the format <*var_name*> within the provider file. When specifying in the genParmStruct, the key should be *var_name*.

The requested operation object may direct the details of the request via the requestParameters object. Additionally, the response, including the headers and body, may be directed via the responseResults object.

The requestParameters object may hold an entry with “mechanism”: “HEADER” to add the header described by the “descriptor” key. A mechanism of “MESSAGE_BODY” may be used to request that the body to be sent come from the <GDK_DATA> descriptor, as well. A mechanism of “URL_PARM” may be used to request that the url be modified with a query as defined by the “descriptor” key.

The responseResults object may hold an entry with a mechanism of “HEADER” to indicate that a specific response header name be processed according to the “descriptor” key.

**Restrictions:** None.

**Authorization (or Privilege required):** User

**Related Information (or Related Services):** GDKGET, GDKWRITE, GDKDEL, GDKLIST

**Return Value (or Return and reason code):**

The various return code constants are documented in the gdkic header file, found in SYS1.SIEAHDRV.H.

| Code | Constant Name | Explanation |
|---|---|---|
| 0 | GDK_OK | Processing was successful |
| 100 | GDK_UNABLE_TO_READ_KEYFILE | The gdkkeyf.json document was not found in the CDA System directory |
| 102 | GDK_UNABLE_TO_PARSE_KEYFILE | When parsing the gdkkeyf.json document, it is not a valid JSON document. |
| 104 | GDK_CLOUD_PROVIDER_NOT_FOUND | When reading the gdkkeyf.json document, for the requested user, the specified cloudProvider was not found. |
| 105 | GDK_NO_RESOURCE_FOUND_IN_KEYFILE | For the specified UserID, and cloudProvider, no credentials entry for / was found. |
| 110 | GDK_PROVIDER_OPEN_FAILURE | The JSON document for the requested cloudProvider does not exist, or was found, but cannot be opened for READ. |
| 111 | GDK_PROVIDER_SPECIFICATION_INVALID | When parsing the JSON document, it was found to be invalid. |
| 112 | GDK_FEATURE_UNSUPPORTED | An unknown contentType was found in the parameterSet |
| 113 | GDK_BUFFER_TOO_SMALL | When attempting to fill the passed buffer, the amount of data is to big to fit in the buffer. |
| 114 | GDK_AUTH_INIT_FAILURE | When using the AWS4 authentication model, the appropriate entry in the gdkkeyf.json document for the “MVSUserID”, “cloud provider”, and “name” (resource) is missing either the "key”, or the "secretkey" key-value pairs. |
| 116 | GDK_AUTH_APPLY_FAILURE | An error occurred while applying the authorization parameters. |
| 118 | GDK_USER_INFO_NOT_FOUND | When reading the gdkkeyf.json document, the requested user entry was not found. |
| 123 | GDK_CONN_NOT_HTTPS | HTTPS was not specified for httpMethod in the provider file. |
| 124 | GDK_ICSF_ERROR | ICSF returned an error during processing. Refer to the accompanying ICSF return code and reason code. |
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified cloud_provider.json file is empty. |

| | | |
|---|---|---|
| 799 | GDK_UNEXPECTED_ERROR | An unexpected error occurred. Contact IBM L2, and provide the logging output. |
| 800 | GDK_CONNECTION_FAILED | Unable to connect to the host set in the cloud provider JSON definition |
| 801 | GDK_TOOLKIT_FAILED | Error occurred in the z/OS Client Web Enablement Toolkit HWTHRQST call |
| 900 | GDK_OBJECT_NOT_FOUND | HTTP status 404 was returned indicating the object was not found. |
| 901 | GDK_ACCESS_DENIED | HTTP status 403 was returned indicating the user is not authorized to access the resource in the desired manner. |
| 902 | GDK_OPERATION_NOT_SUPPORTED | When parsing the cloud provider JSON document, the GETOBJECT description was not found in the "supportedOperations" array. |
| 903 | GDK_RESPONSE_FORMAT_MISMATCH | The request was successful, but the data returned was in a format that did not match what is expected according to the contentType specified in the responseResults for the action in the provider file. |
| 904 | GDK_REQUEST_FAILED | A bad HTTP status (4xx or 5xx) was returned. |

**Examples:** TBD.

**Other:** None.

**Notes to Information Solutions:** TBD

#### 2.4.6.4 GDKVALD – Validate Provider File

A new section, titled “GDKVALD – Validate a Provider file”, is added after the GDKGEN section. Contents are as follows:

The GDKVALD API will allow callers to request that CDA examine a named provider json file. The caller may pass operation names as well as specific key names on input. The API will verify that the requested operation name exists, and may return the value associated with the requested keys. The intended use of this API is to allow callers to validate that the requested provider file meets the requirements of their expected processing. Additionally, it allows retrieval of values from the provider file.

The GDKVALD API processing can additionally be modified through the optionalParmStruct.

**Purpose (or Function):** Perform validation of operations in a provider file, and additionally return values for requested key names.

**Requirements:** When using the Branch Entry for gdkvald(), a Language Environment must be established before this API is invoked. When called from a Language Environment C program, the environment is already created and available. Otherwise, a PreInitialization environment can be used to create a Language Environment (CEEPIPI). The z/OS Language Environment Programming Guide has more details in the “Using preinitialization services” section.

When linking with SYS1.CSSLIB(GDKCSS), then a Language Environment should already be established. When linking with SYS1.CSSLIB(GDKCSSNL), then a Language Environment does not need to be established as it will be created on entry, and terminated at exit.

**Format:**

```
gdkvald (retCodeAddr,
         cloudProvider,
         valParmStructPtr,
        optionalParmStructPtr);
```

When using Branch Entry, and a Language Environment is not established, the GDKVALDN API may be used. This will create a Language Environment upon entry and terminate the Language Environment upon exit.

**Format:**

```
gdkvaldn (retCodeAddr,
         cloudProvider,
         valParmStructPtr,
        optionalParmStructPtr);
```

**Parameters:**

parameter name description

**retCodeAddr**

Specifies the address of a 4-byte field that the API will place the return code into.

**cloudProvider**

Specifies the address of a pointer to a name of the cloud provider json file that will be used during validation of the requested keys. The name must be null-terminated. When retrieving the Cloud Provider definition corresponding to the cloudProvider specified on the API call, the user’s RACF ID, or UserID from the optionalParms, is used to examine the associated OMVS segment to retrieve the home directory. That home directory is examined for a gdk/providers/ sub-directory. If it exists and the <cloudProvider>.json file is found, it will be examined. If the gdk/providers/ directory does not exist or the <cloudProvider>.json file is not found there, then the CDA System Default directory of /usr/lpp/dfsms/gdk/providers/ will be used.

The application may request a list of all supported providers by using the **GETPROVIDERS** API.

**valParmStructPtr**

Specifies the address of the validateParmsStruct block which contains key names to be validated as well as pointers to buffers to hold the returned values if requested.

validateParmsStruct Control Block Mapping

```
/* ---------------------------------------------------------------- */
/* validateParmsStruct Type definition (GDKVALDservice)             */
/*   - maps the data pointed to by the valParmStructPtr             */
/*     parameter                                                    */
/* ---------------------------------------------------------------- */
```

```
 typedef void * GDK_VAL_VALUE_TYPE; /* Generic pointer to value      */

 typedef struct {
   GDK_OPTION_KEY_TYPE GDK_valOp;   /* Operation name                */
   GDK_OPTION_KEY_TYPE GDK_valKey;  /* Key name pointer              */
   GDK_VAL_VALUE_TYPE  GDK_valValue; /* Value buffer pointer         */
   unsigned short      GDK_valLen;  /* Buffer size/Length used       */
   unsigned char       GDK_valRqst; /* Request options               */
   unsigned char       __GDK_rsrv2; /* Reserved                      */
   unsigned char       GDK_valExists; /* 1 = Key exists              */
   unsigned char       GDK_valType; /* Type of value                 */
   unsigned short      GDK_valErr;  /* Entry in error with RC        */
} GDK_VAL_TYPE;                     /* Single Validation Entry       */

 typedef struct {
   uint32_t GDK_NumOfValTypes;      /* Count of Validation Ents      */
   GDK_VAL_TYPE GDK_ValArray[];     /* Array of GDK_VAL_TYPEs        */
 } GDK_VAL_PARMS_TYPE;              /* Header of structure           */

 #define GDK_VAL_INVALID 0          /* Invalid type                  */
 #define GDK_VAL_STRING 3           /* String type                   */
 #define GDK_VAL_INT    4           /* Integer type                  */
 #define GDK_VAL_BOOL   5           /* Boolean type: 0 is false      */
 #define GDK_VAL_ARRAY  6           /* Array of Entries              */
 #define GDK_VALERR_SIMPLE 8        /* Key is not simple type        */

#define GDK_MAX_HOST_NAME_LEN 508   /* Longest host name 0-term      */
#define GDK_VR_ARR_OK   5           /* Return Array of Values OK     */

 typedef struct {
   unsigned short GDK_vType;        /* Type of Entry                 */
   unsigned short GDK_vLen;         /* Length of Entry               */
   char GDK_vStr[GDK_MAX_HOST_NAME_LEN]; /* Area for String          */
 } GDK_VAL_ARR_ENT;                 /* Array entry description       */

 #define GDK_VA_PREF 0x0097         /* Preferred hostname entry      */
 #define GDK_VA_BACK 0x0082         /* Backup hostname entry         */
```

Within a single GDK_VAL_TYPE entry, the fields have the following meanings and uses:

- GDK_valOp field points to a null terminated string that is the name operation to be searched for in the supportedOperations array. Set to NULL if the first instance of GDK_valKey should be returned. Set to the name of the operation that should be searched inside.
- GDK_valKey field points to a null terminated string that is the key name that should be validated. If only wanting to verify the existence of an operation, this should be set to NULL.
- GDK_valValue should point to a buffer large enough to hold the returned value. When GDK_valKey is NULL, the buffer this points to will not be filled.
- GDK_valLen should be set to the maximum size of the buffer. After the call, this will be set to the actual used size if no error, or zero if an error occurred.
- GDK_valRqst is used to indicate the types of values acceptable when multiple types may be returned. Default is 0, indicating only the default type should be returned. Set to GDK_VR_ARR_OK to indicate that an array of GDK_VAL_ARR_ENT entries is acceptable.
- GDK_valExists – On output is set to 1 to indicate the GDK_valKey was found as expected. It is set to 0 to indicate the GDK_valKey was not found as expected. If GDK_valKey was NULL, it is

set to 1 to indicate that the operation object was found in the supportedOperations array. It is set to 0 if the operation name was not found in the supportedOperations array.

- GDK_valErr – Output field regarding result of processing
  - 0 if no error occurred.
  - 8 if the key was found, but was not a simple type (String, Number, Boolean, Null).
  - 113 if the buffer is too small. GDK_valLen will be set to the needed size.
  - 902 if operation is not found
- GDK_valType – On output is set to a value to indicate the content of GDK_valValue.
  - 0 – Value is invalid. (It doesn't point to anything.)
  - 3 – Value is a null-terminated string
  - 4 – Value is a number. (GDK_valLen is 4 for a 4-byte integer)
  - 5 – Value is a 4-byte Boolean value. 0 means false, 1 means true.
  - 6 – Value is an array of GDK_VAL_ARR_ENT entries.

When GDK_valType indicates an array of entries is placed in the buffer that GDK_valVal points to, each entry in the array is laid out in the fixed format described by the GDK_VAL_ARR_ENT mapping:

- GDK_vType – A 2 byte field indicating the type of entry. Values are:
  - 0x0097 – Preferred host name from host array
  - 0x0082 – Backup host name from host array
- GDK_vLen – A 2-byte field containing the length of string following.
- GDK_vStr - A 508-byte character area containing the null-terminated host name which may be of the format: https://<host_name><:port><,sslKey_ring>.

**optionalParmStructPtr**

Specifies an optional method for a user of this API to provide customized processing not provided by default by the CDA API. The API will specify a pointer to a structure as mapped by the data structure **GDK_OPTIONAL_PARMS_TYPE**. This data structure will contain one or more customized overrides or additions. In general, all string values, including keys and values, must be null terminated when being passed to the APIs.

| Optional Parms | | |
|---|---|---|
| **NAME** | **Type** | **Description** |
| "UserID" | 8-byte char | RACF User ID used to retrieve Cloud security credentials for the GET request. |
| "Use-Config-File" | Character | "false" means that the config.json file should not be read for default configuration values.<br>Any other values mean that the config.json should be used. |
| "log-level" | Character | 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 stdout. The levels in order of low to high severity are listed:<br>"DEBUG" means all logging messages are written to.<br>"INFO" means only INFO and higher severity logging messages are written.<br>"NOTICE" means only NOTICE and higher severity logging messages are written.<br>"WARNING" means only WARNING and higher severity logging messages are written.<br>"ERROR" means only ERROR logging messages are written.<br>"NONE" means no messages are written regardless of severity. |

| | | |
|---|---|---|
| “CDA_Session” | Address | Specifies the address of a 12-byte field that was filled in as part of a gdkinit() CPI call. This indicates a session is current and the config file, keyfile, and provider file will not be read again during this API call. |

**Usage (or Usage Notes):**

When the API is invoked without the session handle, if UserID wasn’t provided in the OptionalParms, the current user’s RACF ID is used to retrieve the applicable CloudProvider definition.

When retrieving the CloudProvider definition corresponding to the cloudProvider specified on the API call, the user’s RACF ID, or UserID from the optionalParms, is used to examine the associated OMVS segment in order to retrieve the home directory. That home directory is examined for a gdk/providers/ sub-directory. If it exists and the <cloudProvider>.json file is found, it will be used as the template to communicate with the cloud provider. If the gdk/providers/ directory does not exist or the provider file is not found there, then the CDA System Default directory of /usr/lpp/dfsms/gdk/providers/ will be used.

If a session handle is passed via the optional parameters, then the config.json file, keyfile, and provider file are not re-read.

If the “host” key is requested, and the host key in the provider has a value that is a JSON object containing a “preferred” list and optionally a “backup” list and GDK_valRqst is not GDK_VR_ARR_OK, then the return value will be a randomly chosen host value from the preferred array. If the size of the return buffer is not big enough to contain all of the values, as many complete values that can fit will be returned and the GDK_valErr will be set to RC 113 indicate there was more data.

**Restrictions:** None.

**Authorization (or Privilege required):** User

**Related Information (or Related Services):** None

**Return Value (or Return and reason code):**

The various return code constants are documented in the gdkic header file, found in SYS1.SIEAHDRV.H.

| Code | Constant Name | Explanation |
|---|---|---|
| 0 | GDK_OK | Processing was successful |
| 104 | GDK_CLOUD_PROVIDER_NOT_FOUND | When reading the gdkkeyf.json document, for the requested user, the specified cloudProvider was not found. |
| 110 | GDK_PROVIDER_OPEN_FAILURE | The JSON document for the requested cloudProvider does not exist, or was found, but cannot be opened for READ. |
| 111 | GDK_PROVIDER_SPECIFICATION_INVALID | When parsing the JSON document, it was found to be invalid. |
| 113 | GDK_BUFFER_TOO_SMALL | When attempting to fill the passed buffer, the amount of data is to big to fit in the buffer. |
| 141 | GDK_PARAMETER_ERROR | One of the parameters on the API call was incorrect. Examine the ERROR log message for details on the parameter in error. |

| | | |
|---|---|---|
| 142 | GDK_EMPTY_PROVIDER_FILE | The specified cloud_provider.json file is empty. |
| 902 | GDK_OPERATION_NOT_SUPPORTED | The requested operation name was not found in the supportedOperations array in the provider file. |

**Examples:** TBD.

**Other:** None.

**Notes to Information Solutions:** TBD

### 2.4.6.5 GDKQUERY – Query available CDA functions

A new section, titled "GDKQUERY – Query CDA function availability", is added after the GDKVALD section. Contents are as follows:

**Purpose (or Function):** Retrieve information about the currently supported CDA functions on the current system.

**Requirements:** When using the Branch Entry, a Language Environment must be established before the gdkquery() API is invoked. When called from a Language Environment C program, the environment is already created and available. Otherwise, a PreInitialization environment can be used to create a Language Environment (CEEPIPI). The z/OS Language Environment Programming Guide has more details in the "Using preinitialization services" section. When using the Branch Entry interface, if no Language Environment exists, the alternate gdkqryn() API may be invoked. This will create and tear down a Language Environment for the duration of the call.

When linking with SYS1.CSSLIB(GDKCSS), then a Language Environment should already be established. When linking with the SYS1.CSSLIB(GDKCSSNL), then a Language Environment will be established and then torn down once processing is completed.

**Format:**

```
gdkquery (retCodeAddr,
          queryBuffer,
          queryBufferLen,
          optionalParmStructPtr);
```

When using Branch Entry, and a Language Environment is not established, the GDKQRYN API may be used. This will create a Language Environment upon entry and terminate the Language Environment upon exit.

**Format:**

```
gdkqryn (retCodeAddr,
         queryBuffer,
         queryBufferLen,
         optionalParmStructPtr);
```

**Parameters:**

parameter name description

**retCodeAddr**

Specifies the address of a 4-byte field that the API will place the return code into.

**queryBuffer**

Specifies the address of a pointer to a buffer that is large enough to hold one of the versions of the GDK_FUNCTION_MAP_Vx control blocks. The GDK_FUNCTION_MAP_V1 block is 1024 bytes in size. .

GDK_FUNCTION_MAP_V1 Control Block

```
/**********************************************************************/
/* GDK_FUNCTION_MAP_V1 control block mapping                          */
/**********************************************************************/

typedef struct {
  char gdkf1_eyec[8];                /* Eye catcher for control block */
  unsigned short gdkfq_ver;          /* Version of the Function Map   */
  char _rsrvd1 char[2];              /* Reserved space                */
  struct{                            /* Flags                         */
    int gdkf1_gdkdel   :1,           /* GDKDEL API                    */
        gdkf1_gdkget   :1,           /* GDKGET API                    */
        gdkf1_gdkwrite :1,           /* GDKWRITE API                  */
        gdkf1_gdklist  :1,           /* GDKLIST API                   */
        gdkf1_gdkkeysr :1,           /* GDKKEYSR API                  */
        gdkf1_gdkkeyad :1,           /* GDKKEYAD API                  */
        gdkf1_gdkkeygr :1,           /* GDKKEYGR API                  */
        gdkf1_gdkkeydl :1,           /* GDKKEYDL API                  */
        gdkf1_gdkinit  :1,           /* GDKINIT API                   */
        gdkf1_gdkterm  :1,           /* GDKTERM API                   */
        gdkf1_gdkgen   :1,           /* GDKGEN API                    */
        gdkf1_gdkvald  :1,           /* GDKVALD API                   */
        gdkf1_gdkquery :1,           /* GDKQUERY API                  */
                   :18;              /* Reserved bits                 */
  } api_flags1;                      /* Supported API flags           */

  struct{                            /* Flags                         */
    int gdkf1_fmeta          :1,     /* Object metadata               */
    int gdkf1_exit_multipart :1,     /* Multipart upload for EXIT     */
    int gdkf1_list_prefix    :1,     /* LIST PREFIX support           */
    int gdkf1_list_delim     :1,     /* LIST Delimiter support        */
                   :28;              /* Reserved bits                 */
  } func_flags1                      /* Supported function flags      */

  char _rsrvdEnd[1004];              /* Reserved space                */
} GDK_FUNCTION_MAP_V1;               /* Version 1 function map        */

#define GDKF_V1 1;                   /* Version 1 function map        */
```

**queryBufferLen**

Specifies the address of a 4-byte unsigned number that is the size of the queryBuffer storage, this must be at least 1024 bytes in size.

**optionalParmStructPtr**

Specifies an optional method for a user of this API to provide customized processing not provided by default by the CDA API. The API will specify a pointer to a structure as mapped by the data structure **GDK_OPTIONAL_PARMS_TYPE**. This data structure will contain one or more customized overrides or additions. In general, all string values, including keys and values, must be null terminated when being passed to the APIs.

| Optional Parms | | |
|---|---|---|
| **NAME** | **Type** | **Description** |
| "Use-Config-File" | Character | "false" means that the config.json file should not be read for default configuration values.<br>Any other values mean that the config.json should be used.<br>On subsequent API calls, the "Use-Config-File" parm is ignored because the default config has been set by the gdkinit() call. |
| "log-level" | Character | This setting sets the default logging level for all subsequent API calls. The default logging level may be overridden for specific API calls via the optional Parms on that call. The default 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 stdout. The levels in order of low to high severity are listed:<br>"DEBUG" means all logging messages are written to.<br>"INFO" means only INFO and higher severity logging messages are written.<br>"NOTICE" means only NOTICE and higher severity logging messages are written.<br>"WARNING" means only WARNING and higher severity logging messages are written.<br>"ERROR" means only ERROR logging messages are written.<br>"NONE" means no messages are written regardless of severity. |

**Usage (or Usage Notes):**

When examining the GDK_FUNCTION_MAP, the version field should be examined to understand which version of the map is being returned.

**Restrictions:** None.

**Authorization (or Privilege required):** User

**Related Information (or Related Services):** GDKMSGTR

**Return Value (or Return and reason code):**

The various return code constants are documented in the gdkic header file, found in SYS1.SIEAHDRV.H.

| Code | Constant Name | Explanation |
|---|---|---|

| | | |
|---|---|---|
| 0 | GDK_OK | Processing was successful |
| 113 | GDK_BUFFER_TOO_SMALL | The passed buffer size is too small to hold even the lowest version of the GDK_FUNCTION_MAP block. |
| 799 | GDK_UNEXPECTED_ERROR | An unexpected error occurred. Contact IBM L2, and provide the logging output. |

# 3 z/OS DFSMSdfp Utilities

SC23-6864-60

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

Under the Control section, the OBJNAME description is updated to add a new paragraph after the bullet list of unacceptable characters as follows:

The OBJNAME 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 OBJNAME 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.

Under the Control section, the LOCNAME description is updated to add a new paragraph after the note regarding the data set attribute considerations as follows:

The LOCNAME DD may contain the absolute path of a UNIX directory. When specified with the UPLOAD command, this indicates that all non-hidden regular files within the directory tree may be selected for upload to the cloud provider. Symbolic links are followed to examine the target of the link. When specified with the DOWNLOAD command, this indicates that all objects downloaded should be placed in that directory. If any pseudo-directory characters (forward slash /) are found in the object name, then a subdirectory with that name will be attempted to be created.

Note:

z/OS UNIX file names may contain unprintable characters that will not be url-encoded, resulting in a failure to upload that file. Not all valid z/OS UNIX file names can be used as cloud object names. Some characters may be deemed unacceptable by the cloud provider, and may result in a failure even if url-encoded as specified in the urlEncodeChars key:value pair in the cloud provider file.

Under the Control section, add a new section titled BUCKET as follows:

BUCKET Statement

The BUCKET statement describes the cloud bucket for a multi object operation. A multi object operation may be:

- An operation where multiple z/OS UNIX files are uploaded to a bucket.
- An operation where multiple cloud objects are downloaded to a z/OS UNIX directory
- An operation where multiple cloud objects are deleted from a bucket.

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. If the UPLOAD command is used where the LOCNAME describes a UNIX directory, this bucket name is where the objects will be placed.

The object names may be prefixed with the value of the PREFIX keyword as well if specified. If the DOWNLOAD command is used, the bucket name is the bucket where object names are listed. The PREFIX keyword may also adjust the object names that are considered for processing.

The first non-blank record in a data set, member, or in-stream data, or non-blank line in a UNIX file is read from the DD and the content is used as the bucket name. The bucket name must start with a forward slash, followed by the bucket or container name ending in a forward slash. The cloud bucket or container must already exist. 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 ("|")

Under the Control section, the SYSIN description is updated to add a new row in the **Command** section of the table after the **List** row as follows:

| | |
|---|---|
| OPERATION(<oper_name>) | Requests the operation named <oper_name> to be performed. <oper_name> is an operation found in the **supportedOperations** array in the provider file. minimum length: OPER(<oper_name>) |

Additionally, new rows are added to the **Keywords** section as follows:

| | |
|---|---|
| PREFIX(<list_prefix>) | Requests that only object names that begin with <list_prefix> are returned for the LIST command. <list_prefix> is a string of characters. The provider file must include the URL_PARM requestParameter definition for “prefix”. minimum length: PRE(<list_prefix>)<br>Requests that only bucket names that begin with <list_prefix> are returned for the LIST command when the object name is only /.<br><br>When specified with the UPLOAD command and the LOCNAME is for a directory, each object name created in the cloud provider will contain the requested prefix.<br><br>When specified with the DOWNLOAD or DELETE command the list of objects considered for processing is modified with the prefix. |
| DELIM(<single_char>) | Requests that only the portion of the object name that is common up to the <single_char> is returned. This is most commonly used with the forward-slash character (/) to utilize a pseudo directory structure to |

| | |
|---|---|
| | object names. The provider file must include the URL_PARM requestParameter for “delimiter”. |
| REGEX(<reg_expr>) | Indicates that the command should process multiple things, and selection of the object name or local z/OS UNIX file should be done according to whether it matches the <reg_expr>. Only applies for UPLOAD where LOCNAME indicates a UNIX directory, DOWNLOAD or DELETE where OBJNAME indicates a bucket name, |
| TEST | Indicates that processing should not be performed for a multiple object command. Only selection is performed. This can be used to determine what would be processed in a multiple object command. For example, when the UPLOAD command is specified with TEST and the LOCNAME indicates a z/OS UNIX directory, messages will be issued indicating the source z/OS UNIX file name, and target cloud object name after applying the prefix if PREFIX was specified, and applying the REGEX regular expression to select a file for upload processing. |

New examples are added to the **GDKUTIL Examples** section as follows:

**Example 3: Create a Bucket**

In this example, the OPERATION command is used to create a new bucket in the cloud provider.

```
//CRBUCKET EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
 OPERATION(CREATEBUCKET) PROVIDER(IBMCOS)
/*
//OBJNAME DD *
  /newbucket05/
/*
```

**Example 4: List Objects in a cloud storage bucket with filtering**

In this example, the GDKUTIL utility is used to list the objects within a specific bucket. The PREFIX keyword is used to filter the results to only those object names that begin with the prefix, “images/”. The DELIM keyword is used to further filter those results to only the portion that has a forward slash next. Some applications like to use a forward slash character in the object name as a pseudo-directory character in order to group objects by ‘directory’ order.

```
//LISTDIR1 EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
 LIST PROVIDER(IBMCOS)
 PREFIX(2023-jan/webapp/) DELIM(/)
/*
//OBJNAME DD *
  /appimages05/
/*
```

**Example 5: Upload multiple files to cloud storage**

In this example, the GDKUTIL utility is used to upload an entire directory tree of files to objects in cloud storage. The LOCNAME DD indicates that the /u/user01/images/ directory is the place to start looking for files to upload. The PREFIX keyword is used here to request that every cloud object begin with /cloudbucket01/user01/images/ . Individual files found will have their path name relative to the LOCNAME value appended to /cloudbucket01/user01/images/ . The REGEX keyword is used here to only select filenames that end in ".png".

```
//UPMULTI EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 UPLOAD PROVIDER(IBMCOS)
 PREFIX(user01/images) REGEX(.*\.png)
/*
//OBJNAME DD *
  /cloudbucket01/
/*
//LOCNAME DD *
  /u/user01/images/
/*
```

**Example 6: Download multiple objects from cloud storage**

In this example, the GDKUTIL utility is used to download multiple objects from the cloud provider described in the IBMCOS.json provider file. The CONVERT keyword requests translation of each file from UTF-8 to EBCDIC. The LOCNAME DD describes the target directory to place the downloaded objects into. The OBJNAME DD specifies the bucket name where the objects are to be found. The PREFIX keyword limits the selection of object names to only the ones that begin with webapp/logs/. The REGEX keyword further filters that list so that only objects with names ending in ".txt" are selected for download processing. Pseudo-directory characters (forward slash /) will cause the object to be placed in the z/OS UNIX directory with the same name.

```
//DOWNMULT EXEC PGM=GDKUTIL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 DOWNLOAD PROVIDER(IBMCOS) CONVERT
 PREFIX(webapp/logs/) REGEX(.*\.txt)
/*
//OBJNAME DD *
  /cloudbucket01/
/*
//LOCNAME DD *
  /u/appproc01/logs/
/*
```

**Example 7: Delete multiple objects from cloud storage**

In this example, the GDKUTIL utility is used to delete multiple objects from the cloud provider described in the IBMCOS.json provider file. The OBJNAME DD describes the bucket name containing the objects to be deleted. The PREFIX keyword limits the selection of object names to only the ones that begin with webapp/logs/. The REGEX keyword further filters that list so that only objects with names ending in ".log" are selected for deletion.

```
//DELMULT  EXEC PGM=GDKUTIL,REGION=0M
```

```
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
 DELETE PROVIDER(IBMCOS)
 PREFIX(webapp/logs/) REGEX(.*\.log)
/*
//OBJNAME DD *
  /cloudbucket01/
/*
```

# 4 z/OS MVS System Messages: Vol 5 (EDG-GLZ)

SA38-0672-60

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

## 4.1 GDKU0023E MISSING OPERATION NAME

GDKU0023E MISSING OPERATION NAME: OPERATION(name)

**Explanation**

The OPERATION command was specified on the SYSIN, but the required sub-parameter of an operation name was not included within open and close parentheses.

**System action**

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

**Operator response**

None.

**System programmer response**

Ensure that the OPERATION command contains a name within the parentheses.

**Source**

DFSMSdfp CDA

## 4.2 GDKU0025E UNSUPPORTED MULTI OPERATION

GDKU0025E UNSUPPORTED MULTI OPERATION

**Explanation**

The requested command does not support processing multiple things.

**System action**

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

**Operator response**

None.

**System programmer response**

Ensure that the requested command can perform a multi-object operation. An UPLOAD, DELETE or DOWNLOAD multi-object operation requires that the OBJNAME DD specifies only a bucket name.

**Source**

DFSMSdfp CDA

## 4.3 GDKU0026E MULTI OBJECT REQUEST

GDKU0026E MULTI OBJECT REQUEST REQUIRES ONLY A BUCKET NAME FOR OBJNAME: <obj_name>

**Explanation**

A command requesting processing of multiple objects was used. However the OBJNAME DD specified more than just a bucket name.

**System action**

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

**Operator response**

None.

**System programmer response**

The PREFIX keyword, REGEX keyword, or both should be used to narrow the search and selection parameters for the objects in the cloud bucket.

**Source**

DFSMSdfp CDA

## 4.4 GDKU0106I LOCNAME <loc_name> OBJNAME <obj_name>

GDKU0106I LOCNAME <loc_name> OBJNAME <obj_name>

**Explanation**

During a multiple object processing request, processing was successful. The cloud object name is displayed by <obj_name>. The local z/OS name is displayed by <loc_name>

**System action**

None.

**Operator response**

None.

**System programmer response**

None.

**Source**

DFSMSdfp CDA

## 4.5 GDKU0107W FILENAME <file_name> SKIPPED

GDKU0107W FILENAME <file_name> SKIPPED BECAUSE OBJECT NAME <object_name> WOULD BE TOO LONG

**Explanation**

During a multiple object download, the constructed local name based on the target directory and object name portion would be too long for the UNIX filesystem.

**System action**

The task continues. The return code is 4

**Operator response**

None.

**System programmer response**

Restrict the target directory in the LOCNAME DD or object name portion via the PREFIX keyword.

**Source**

DFSMSdfp CDA

## 4.6 GDKU0108E NO OBJECTS PROCESSED

GDKU0108E NO OBJECTS PROCESSED

**Explanation**

During a multiple object processing request, nothing was processed.

**System action**

The task was performed. The return code is 8.

**Operator response**

None.

**System programmer response**

None.

**Source**

DFSMSdfp CDA

## 4.7 GDKU0109E UNABLE TO PROCESS <obj_name>

GDKU0109E UNABLE TO PROCESS <obj_name> DUE TO MKDIR ERROR FOR <dir_name> RC: <rc> - <err_text>

**Explanation**

During a multi-object download to a z/OS UNIX directory, a pseudo directory character (forward slash / ) was found in the object name, resulting in the attempt to create a directory with that name. The directory creation of <dir_name> failed with <rc> and explanation text <err_text>.

**System action**

Download processing of <obj_name> is not done. Overall return code is 8. Processing continues.

**Operator response**

None.

**System programmer response**

Determine if there is a way to correct the error explained in <err_text> and run the command again.

**Source**

DFSMSdfp CDA

## 4.8 GDKU0110I PROCESSED <count> OBJECTS

GDKU0110I PROCESSED <count> OBJECTS

**Explanation**

After completion of a multiple object request, the total count of objects processed is displayed.

**System action**

None

**Operator response**

None.

**System programmer response**

None.

**Source**

DFSMSdfp CDA

## 4.9 GDKU0111I DELETED OBJNAME

GDKU0111I DELETED OBJNAME <obj_name>

**Explanation**

During a multiple object delete request, <obj_name> was successfully deleted.

**System action**

None.

**Operator response**

None.

**System programmer response**

None

**Source**

DFSMSdfp CDA

## 4.10 GDKU0112E ERROR RC: <rc> PROCESSING

GDKU0112E ERROR RC: <rc> PROCESSING <loc_name>

**Explanation**

During a multiple object upload, processing of <loc_name> failed with CDA return code <rc>. Examine the return code to understand the specific error the z/OS file or data set encountered.

**System action**

The <loc_name> item is not processed. Return code is 8. Processing continues.

**Operator response**

None.

**System programmer response**

Fix the error and resubmit the job.

**Source**

DFSMSdfp CDA

## 4.11 GDKU0113I TEST MODE SELECTED LOCNAME

GDKU0113I TEST MODE SELECTED LOCNAME <loc_name> OBJNAME <obj_name>

**Explanation**

The TEST keyword was specified for a multi-object command. Selection processing was done for the command including any regular expression matching if the REGEX keyword was specified. Selected z/OS UNIX files or cloud object names are displayed.

**System action**

The <loc_name> or <obj_name> item is not processed. Return code is 0. Processing continues.

**Operator response**

None.

**System programmer response**

None

**Source**

DFSMSdfp CDA

## 4.12 GDKU0114I RESPONSE BODY CONTENTS

GDKU0114I RESPONSE BODY CONTENTS <respBody>

**Explanation**

An HTTP error was encountered during processing as indicated on the GDKU0101E message. The contents of the error response body from the cloud object server is displayed in the <respBody> area.

**System action**

None.

**Operator response**

None.

**System programmer response**

Examine the contents of the <respBody> for more information about what error the cloud object server returned.

**Source**

DFSMSdfp CDA

## 4.13 GDKU0115I FILENAME <file_name> IS UNSUPPORTED

GDKU0115I FILENAME <file_name> IS UNSUPPORTED

**Explanation**

During multi object upload processing of files within a z/OS directory, an entry was found that is not a regular z/OS UNIX file. This file type is not supported for upload to cloud object storage.

**System action**

None. Processing continues.

**Operator response**

None.

**System programmer response**

Examine the type of the named <file_name>.

**Source**

DFSMSdfp CDA

## 4.14 GDKU0150W UNABLE TO OPEN DIRECTORY

GDKU0150W UNABLE TO OPEN DIRECTORY <dir_name>: <error_info>

**Explanation**

A directory name was specified as the LOCNAME value for a multi object upload or download, but it could not be opened. <dir_name> indicates the specified directory name. <error_info> indicates the formatted error text from the system.

**System action**

The named directory is not processed. Processing continues with return code 4.

**Operator response**

None.

**System programmer response**

Examine the specified name and ensure that directory exists and ends with a forward slash character.

**Source**

DFSMSdfp CDA

## 4.15 GDKU0151W UNABLE TO RETRIEVE INFO

GDKU0151W UNABLE TO RETRIEVE INFO FOR: <loc_name>: <error_info>

**Explanation**

Information about a local UNIX file or data set could not be retrieved. <loc_name> indicates the name of the UNIX file or data set. <error_info> indicates the formatted error text from the system.

**System action**

The named <loc_name> is not processed. Processing continues with return code 4.

**Operator response**

None.

**System programmer response**

Examine the named file or data set and ensure it is accessible.

**Source**

DFSMSdfp CDA

## 4.16 GDKU0152E ERROR COMPILING SPECIFIED REGULAR EXPRESSION

GDKU0152E ERROR COMPILING SPECIFIED REGULAR EXPRESSION <regex>. RC <rc> - <errtext>

**Explanation**

A regular expression was specified on the REGEX keyword. An error occurred compiling the regular expression for later use matching. <regex> is the specified regular expression. <errtext> is the error text from the regerror() call.

**System action**

Processing stops with return code 8.

**Operator response**

None.

**System programmer response**

Examine the error text and resolve the problem with the specified regular expression.

**Source**

DFSMSdfp CDA