lccGoogleAPI
Manual

Contents

Description

This document describes how to use the lccGoogleAPI.

lccGoogleAPI
is a Command Line program created to perform tasks through Google. At this time, these tasks include:

- Create users (also from lists)
- Modify users (also from lists)
- Delete users (also from lists)
- List user
- List all users
- List user messages
- List organizations
- Create organization
- Modify organization (due to bug in Google's API infrastructure, we recommend not using this module)
- Create group
- List group
- Modify group
- Delete group
- Group add member
- Group remove member
- Group add multiple members
- Group remove multiple members
- List group members

The program can also encrypt/decrypt files (like Request Files).

On 'List user/all users', you can specify any/all of the 61 attributes Google provides, i.e. GivenName, Primary Email, etc.

The program receives its instructions from a Logic File. It can run in Run-Once mode, or Back End Server mode.

In Run-Once mode, you supply a Logic File, it runs against the settings provided and then stops.

In Back End Server mode, it continually checks a 'Requests' folder for new requests. It then performs requests as they come in, and archives those requests once completed.

The program provides Logic File – Keys for defining Completed and Failed tasks, logs, etc.

The lccGoogleAPI has been coded against Google's latest standards (as of April 2014), which include:

- OAuth2
- Google Admin SDK
  ref: https://developers.google.com/admin-sdk

Due to bug in Google's API infrastructure, we recommend not using the Modify Organization module. The causes the modfiy settings to apply to the parent org, even though the sub-org id is provided. As of 20210819.

Requirements:
Google's latest processes require a certificate created/downloaded from their Google Apps Admin Console, ref: https://admin.google.com/AdminHome
This certificate allows you to perform tasks on your Google Domain/Accounts without providing a Log-in/Password.

To get this certificate/use, you will need the following:

- an account/domain set-up with Google for Gmail
- an account with admin rights in the Google App Admin Console
- create an API Service Account through the Google Developers Console, ref: https://console.developers.google.com/project
- download the OAuth Key file (.p12) assigned to that Service Account
- place the .p12 file with your program and configure the Logic File (see below) to point to that .p12 file

The following libraries are included with the installation ZIP and are required to be in the same location as the program:

- Google.Apis.Admin.Directory.directory_v1.dll
- Google.Apis.Auth.dll
- Google.Apis.Auth.PlatformServices.dll
- Google.Apis.Core.dll
- Google.Apis.Discovery.v1.dll
- Google.Apis.dll
- Google.Apis.Oauth2.v2.dll
- Google.Apis.PlatformServices.dll
- Newtonsoft.Json.dll

Installation Steps

- copy the lccGoogleAPI.exe and all .dll files to a folder
- create a Logic File, depending on the need, i.e. Back End, etc.
  Folder Structure
  - create folders for the values supplied on Keys:
    - lcc:requestsPath
      - ex: .\requests
      - in this folder, also created a sub-folder: completed
    - lcc:responsesPath
      - ex: .\responses
    - lcc:requestsListsPath
      - ex: .\requests\lists
    - lcc:requestsToFilePath
      - ex: .\requests\toFile
    - lcc:indexesPath
      - ex: .\requests\indexes
- run the lccGoogleAPI.exe with the supplied Logic File name

Logic/Request File Description/Syntax

The Logic File is a Tab delimited text file. Any lines not recognized as a valid Key/Value pair, will be ignore and can be used as remarks/other.

There are two categories of Keys provided through the Logic File:

- General program settings
  - these keys provide settings to the program for running and all requests.
- Requests
  - these keys provide settings per request.

Requests can be embedded directly into the Logic File or contained in their own file(s).

The Logic File uses the syntax:

Syntax: **[Key]** *[tab]* **[Value]**
Example: lcc:key value

Logic File – General Setting Keys

A Logic File can reference another Logic File. This allows you to organize your settings into 'global' files. For example, you could place your general, encryption, smtp, paths, etc. settings into different Logic Files and then reference those from the main one. To reference another Logic File, use this syntax:

Syntax: **lcc:logicAdhocPath** *[tab]* **[path]**
Example: lcc:logicAdhocPath lccGoogleAPI-global-SMTP-logic.txt
Example #2 (multiple):
lcc:logicAdhocPath lccGoogleAPI-global-SMTP-logic.txt
lcc:logicAdhocPath lccGoogleAPI-global-timers-logic.txt

lcc:logPath *(optional)(none to one)*

If you want the program to log information/steps to a log file, provide the path here. The path should include the full path, plus the File Root name. The program will automatically append the Year+Month+".log" to the filename, aka filename-201404.log.

Syntax: **lcc:logPath** *[tab]* **[path]**
Example: lcc:logPath e:\folder\logs\lccAuditPasswordsLog

lcc:debugLevel *(optional)(none to many)*

Controls what information is provided during processing.

Provide this key for each Debug Level you wish to enable.
Debug Levels:
1 Show Locked Requests.

Syntax: **lcc:debugLevel** *[tab]* **[...]**
Example: lcc:debugLevel 1
Example #2 (multiple):
lcc:debugLevel 1
lcc:debugLevel 2

lcc:requestsListsDelimeter *(optional)(one per Logic File)*

If a 'list' is provided to a request, how is that list's columns delimited. Provide the actual character, i.e. ',', or '[tab]' for the tab character.
Syntax: **lcc:requestsListsDelimiter** *[tab]* **[...]**
Example: lcc:requestsListsDelimiter [tab]

lcc:requestResponseFromGoogleRetries *(optional)(one per Logic File)*

How many retries should be attempted if a request to Google fails.

Syntax: **lcc:requestResponseFromGoogleRetries** *[tab]* **[#]**
Example: lcc:requestResponseFromGoogleRetries 3

lcc:propercase *(optional)(one per Logic File)*
Force 'proper case' format for First, Last names. This will change the first character to an uppercase and all other letters to lowercase. The only valid value is 'YES'.

Syntax: **lcc:propercase** *[tab]* **[YES]**
Example: lcc:propercase YES

lcc:sourceEncryptedPasswordHash *(optional)(one per Logic File)*

Hash value supplied for encryption. Must be used with keys 'lcc:sourceEncryptedSaltKey' and 'lcc:sourceEncryptedVIKey'. Only used if at least one of the keys 'lcc:encryptArchivedRequests' or 'lcc:encryptArchivedLists' are used.

Any set of characters can be used.

Syntax: **lcc:sourceEncryptedPasswordHash** *[tab]* **[...]**
Example: lcc:sourceEncryptedPasswordHash s0m3C00lPhr@$e

lcc:sourceEncryptedSaltKey *(optional)(one per Logic File)*

Salt Key value supplied for encryption. Must be used with keys 'lcc:sourceEncryptedPasswordHash' and 'lcc:sourceEncryptedVIKey'. Only used if at least one of the keys 'lcc:encryptArchivedRequests' or 'lcc:encryptArchivedLists' are used.

Any set of characters can be used.

Syntax: **lcc:sourceEncryptedSaltKey** *[tab]* **[...]**
Example: lcc:sourceEncryptedSaltKey s0m3C00lPhr@$e

lcc:sourceEncryptedVIKey *(optional)(one per Logic File)*

VI Key value supplied for encryption. Must be used with keys 'lcc:sourceEncryptedPasswordHash' and 'lcc:sourceEncryptedSaltKey'. Only used if at least one of the keys 'lcc:encryptArchivedRequests' or 'lcc:encryptArchivedLists' are used.

Any set of characters can be used, but, must be exactly 16 characters long.

Syntax: **lcc:sourceEncryptedVIKey** *[tab]* **[...]**
Example: lcc:sourceEncryptedVIKey s0m3C00lPhr@$e12

lcc:encryptArchivedRequests *(optional)(one per Logic File)*

If supplied, will encrypt request file before archiving. The only valid value is 'YES'.

Syntax: **lcc:encryptArchivedRequests** *[tab]* **[YES]**
Example: lcc:encryptArchivedRequests YES

lcc:encryptArchivedLists *(optional)(one per Logic File)*

If supplied, will encrypt request lists before archiving. The only valid value is 'YES'.

Syntax: **lcc:encryptArchivedLists** *[tab]* **[YES]**

Example: lcc:encryptArchivedLists YES

lcc:SMTPMailServer *(optional)(one per Logic File)*

Supplies the SMTP server for sending email notifications.

Syntax: **lcc:encryptArchivedLists** *[tab]* **[YES]**
Example: lcc:encryptArchivedLists YES

lcc:SMTPNotificationsTo *(optional)(none to many per Logic File)*

Supplies any emails that should receive notifications.

Syntax: **lcc:SMTPNotificationsTo** *[tab]* **[...]**
Example: lcc:SMTPNotificationsTo youremail@college.edu
Example #2: (multiple emails)
lcc:SMTPNotificationsTo youremail@college.edu
lcc:SMTPNotificationsTo anotherperson@college.edu

lcc:SMTPNotificationsFrom *(optional)(one per Logic File)*

Supplies the 'from' email when sending email notifications.

Syntax: **lcc:SMTPNotificationsFrom** *[tab]* **[...]**
Example: lcc:SMTPNotificationsFrom ournotifications@college.edu

lcc:SMTPNotificationsSubject *(optional)(one per Logic File)*

Supplies the 'subject' for emails when sending email notifications.

Syntax: **lcc:SMTPNotificationsSubject** *[tab]* **[...]**
Example: lcc:SMTPNotificationsSubject A Notification For You

lcc:SMTPNotificationsSingleSendPath *(optional)(one per Logic File)*

Supplies the path where to create 'single send' files. A single send file is created when a notification is sent. No further notifications will be emailed until either the file is deleted, or the date changes. This provides an anti-spam process to prevent your recipients being spammed when there are more than one notification (error) in a day.

Syntax: **lcc:SMTPNotificationsSingleSendPath** *[tab]* **[…]**
Example: lcc:SMTPNotificationsSingleSendPath .\singleSend.txt

lcc:abortProgramPath *(optional)(one per Logic File)*

Supplies the path where an Abort File is checked for. An abort file is any file that you use an 'abort' flag to the program. If the file exists and contains any content, the program (if running in Back End mode) will abort and delete the abort file. The contents of the file do not matter.

Syntax: **lcc:abortProgramPath** *[tab]* **[…]**
Example: lcc:abortProgramPath .\abortMe.txt

lcc:requestsPath *(mandatory – if Back End)(one per Logic File)*

Supplies the path where Request files should be found. If any files with the extension of '.new' are found, they will be processed, then archived.

Syntax: **lcc:requestsPath** *[tab]* **[…]**
Example: lcc:requestsPath .\requests

lcc:responsesPath *(mandatory – if Back End)(one per Logic File)*

Supplies the path where Responses files should be created. At this time, this Key is not being used, as no need exists.

Syntax: **lcc:responsesPath** *[tab]* **[…]**
Example: lcc:responsesPath .\responses

lcc:requestsListsPath *(mandatory – if Back End)(one per Logic File)*

Supplies the path where Request Lists should be archived after they are processed. Lists are found in the same folder as the Requests, but, moved to this path when finished.

Syntax: **lcc:requestsListsPath** *[tab]* **[…]**
Example: lcc:requestsListsPath .\requests\lists

lcc:requestsToFilePath *(optional)(one per Logic File)*

If Request(s) ask the program to create new Request To Files (using key 'lcc:requestsToFile' (instead of processing them), they will be stored in this path. This is beneficial if you want to create a requests for use later, i.e. copy/move the requests in the requests path later.

Syntax: **lcc:requestsToFilePath** *[tab]* **[…]**
Example: lcc:requestsToFilePath .\requests\toFile

lcc:indexesPath *(mandatory – if Back End)(one per Logic File)*

When requests are completed, indexe(s) will be created listing which User Ids were either Created, Modified, or Deleted.

Syntax: **lcc:indexesPath** *[tab]* **[…]**
Example: lcc:indexesPath .\requests\toFile

lcc:googleServiceAccountClientId *(mandatory)(one per Logic File)*

Google Service Account Client Id used to connect to the Google API.

Syntax: **lcc:googleServiceAccountClientId** *[tab]* **[…]**
Example: lcc:googleServiceAccountClientId 1234.apps.googleusercontent.com

lcc:googleServiceAccountEmail *(mandatory)(one per Logic File)*

Google Service Account Email used to connect to the Google API.

Syntax: **lcc:googleServiceAccountEmail** *[tab]* **[...]**
Example: lcc:googleServiceAccountEmail 1234@developer.gserviceaccount.com

lcc:googlePrivateKeyPath *(mandatory)(one per Logic File)*

Google Certificate Key file path.

Syntax: **lcc:googlePrivateKeyPath** *[tab]* **[...]**
Example: lcc:googlePrivateKeyPath .\keys\ourKeyFile.p12

lcc:googlePrivateKeyPassword *(mandatory)(one per Logic File)*

Google Password used to connect to the Google API.

Syntax: **lcc:googlePrivateKeyPassword** *[tab]* **[...]**
Example: lcc:googlePrivateKeyPassword notasecret

lcc:googleAPIApplicationName *(mandatory)(one per Logic File)*

Application Name provide to the Google API as the connecting program. The name doesn't matter and is only used in Google Logging to identify the program making the request(s).

Syntax: **lcc:googleAPIApplicationName** *[tab]* **[...]**
Example: lcc:googleAPIApplicationName lccGoogleAPI

lcc:googleAPIImpersonateId *(mandatory)(one per Logic File)*

What 'admin' account to impersonate when doing tasks.

Syntax: **lcc:googleAPIImpersonateId** *[tab]* **[...]**
Example: lcc:googleAPIImpersonateId ourAdmin@my.college.edu

lcc:googleAPIScope *(mandatory)(at least one per Logic File)*

What 'scopes' you are allowing the program to perform within Google.

Provide this key for each scope you want used.

These values initialize the scopes defined by Google at:
https://developers.google.com/identity/protocols/googlescopes

Valid Values

- AdminDirectoryUser
- AdminDirectoryGroup
- AdminDirectoryOrgUnit
- GmailReadOnly
- MailGoogleCom
- AdminDirectoryOrgUnit
- [if none of the above used, will add the value as supplied, ex: "https://www.googleapis.com/auth/admin.directory.orgunit"]

What 'admin' account to impersonate when doing tasks.
Syntax: **lcc:googleAPIScope** *[tab]* **[...]**
Example: lcc:googleAPIScope AdminDirectoryUser
Example (#2, multiple): lcc:googleAPIScope AdminDirectoryUser
lcc:googleAPIScope AdminDirectoryGroup
https://www.googleapis.com/auth/admin.directory.orgunit

lcc:multipleRequestsDelay *(optional)(one per Logic File)*

If multiple requests are found, what delay (in milliseconds) should happen between requests. i.e. 1000 = 1 second

Syntax: **lcc:multipleRequestsDelay** *[tab]* **[#]**

Example: lcc:multipleRequestsDelay 1000

lcc:requestResponseFromGoogleDelay *(optional)(one per Logic File)*

How long to wait on retries if request to Google fails. i.e. 1000 = 1 second.

It is highly recommend to have a delay between requests, since Google API does take some time to create/etc. objects. For example, when 'Create User' (or create group, organizatoin, etc.) is used, this program will query to see if that user already exists, then create the user and finally requery for that same user to see if it was created. Since this program can do all that in a couple milliseconds, Google doesn't have enough time to create and update its own directory/schema. A 1000 (ms) delay seems a good balance.

Syntax: **lcc:requestResponseFromGoogleDelay** *[tab]* **[#]**
Example: lcc:requestResponseFromGoogleDelay 1000

lcc:heartbeatSeconds *(optional)(one per Logic File)*

How often should a 'heartbeat' happen. A 'heartbeat' is just a status displayed to the screen/log that the program is still alive. The value is in seconds. i.e. 60 = 1 minute

Syntax: **lcc:heartbeatSeconds** *[tab]* **[#]**
Example: lcc:heartbeatSeconds 60

lcc:requestsCheckSeconds *(optional)(one per Logic File)*

How often should the program check for new requests. The value is in seconds. i.e. 3 = 3 seconds.

Syntax: **lcc:requestsCheckSeconds** *[tab]* **[#]**
Example: lcc:requestsCheckSeconds 3

lcc:encryptFileSourcePath *(optional)(one per Logic File)*

If provided, will encrypt the file at this path. Needs to be used with the Key 'lcc:encryptFileTargetPath'.

Syntax: **lcc:encryptFileSourcePath** *[tab]* **[...]**

Example: lcc:encryptFileSourcePath .\encryptMe.txt

lcc:encryptFileTargetPath *(optional)(one per Logic File)*

If provided, will encrypt a file to this path. Needs to be used with the Key 'lcc:encryptFileSourcePath'.

Syntax: **lcc:encryptFileTargetPath** *[tab]* **[…]**
Example: lcc:encryptFileTargetPath .\meEncrypted.txt

lcc:decryptFileSourcePath *(optional)(one per Logic File)*

If provided, will decrypt the file at this path. Needs to be used with the Key 'lcc:decryptFileTargetPath'.

Syntax: **lcc:decryptFileSourcePath** *[tab]* **[…]**
Example: lcc:decryptFileSourcePath .\decryptMe.txt

lcc:decryptFileTargetPath *(optional)(one per Logic File)*

If provided, will decrypt a file to this path. Needs to be used with the Key 'lcc:decryptFileSourcePath'.

Syntax: **lcc:decryptFileTargetPath** *[tab]* **[…]**
Example: lcc:decryptFileTargetPath .\meDecrypted.txt

Logic/Request File – Request Setting Keys

A Logic/Request File can contain one to many requests.

Each request is encompassed between a 'lcc:requestBegin' and 'lcc:requestEnd' Key.

The order of the keys does not matter, except that the 'lcc:requestBegin' must be the first settings for a request, and 'lcc:requestEnd' must be the last setting for a request.

Some keys are only detailed in the Request Logic Examples section, as they format/example is self explanatory.

The following 'type' of requests are supported:

- Create User
- Modify User
- Delete User
- Create Users From List
- Modify Users From List
- Delete Users From List
- List user
- List all users
- List user messages
- List organizations
- Create Organization
- Modify Organization
- Create Group
- List Group
- Modify Group
- Delete Group
- Group Add Member
- Group Remove Member
- Group Add Multiple Members
- List Group Members

lcc:requestBegin *(mandatory)(one per request)*

The is the beginning of a request. Must be paired with a 'lcc:requestEnd' key. The only valid value is 'YES'.

Syntax: **lcc:requestBegin** *[tab]* **[YES]**
Example: lcc:requestBegin YES

lcc:requestEnd *(mandatory)(one per request)*

The is the end of a request. Must be paired with a 'lcc:requestEnd' key. The only valid value is 'YES'.

Syntax: **lcc:requestEnd** *[tab]* **[YES]**

Example: lcc:requestEnd YES

lcc:requestSkip *(optional)(one per request)*

If provided with a request, the request will be ignored. This is beneficial if you want to have a request defined, but, not used. The only valid value is 'YES'.

Syntax: **lcc:requestSkip** *[tab]* **[YES]**
Example: lcc:requestSkip YES

lcc:requestId *(mandatory)(one per request)*

Provides an 'id' to be used in logging. This id can be used as a reference when researching previously processed requests. The 'id' can be any set of characters.

Syntax: **lcc:requestId** *[tab]* **[...]**
Example: lcc:requestId 2014043-task123

lcc:requestType *(mandatory)(one per request)*

Provides the 'type' of request. The type defines what the request can do.

Syntax: **lcc:requestType** *[tab]* **[...]**
Example: lcc:requestType Create User

lcc:requestUserId *(mandatory)(one to many per request)*

The account's 'user id' that is being processed.

You can supply more than one of these keys if you have mutliple user ids.

Also see lcc:requestUserIdFromFile to load ids from a file path.

Syntax: **lcc:requestUserId** *[tab]* **[...]**
Example: lcc:requestUserId someuser@my.college.edu

lcc:requestUserIdFromPath *(mandatory)(one to many per request)*

Supply account user id(s). This will load a user id for each line in the path provided. Acts the same as if you provided lcc:requestUserId for each line.

Syntax: **lcc:requestUserIdFromPath** *[tab]* **[...]**
Example: lcc:requestUserIdFromPath \\server\share$\folder\ourUsers.txt

lcc:requestUserGivenName *(mandatory/optional)(one per request)*

The account's 'first name' that is being processed.

Syntax: **lcc:requestUserGivenName** *[tab]* **[...]**
Example: lcc:requestUserGivenName John

lcc:requestUserFamilyName *(mandatory/optional)(one per request)*

The account's 'last name' that is being processed.

Syntax: **lcc:requestUserFamilyName** *[tab]* **[...]**
Example: lcc:requestUserFamilyName Doe

lcc:requestUserFullName *(mandatory/optional)(one per request)*

The account's 'full name' (display) that is being processed.

Note: Google only allows Read-Only on this field at this time, which they (Google) sets as the concatenation of the Given Name (first) and Family Name (last).

Syntax: **lcc:requestUserFullName** *[tab]* **[...]**
Example: lcc:requestUserFullName Doe, John

lcc:requestUserPassword *(mandatory/optional)(one per request)*

The account's 'password' that is being processed.

Syntax: **lcc:requestUserPassword** *[tab]* **[…]**
Example: lcc:requestUserPassword S0m3C00lP@ssw0rd

lcc:requestsToFile *(optional)(one per request)*

If supplied, will create a request file in the 'lcc:requestsToFilePath' path. Instead of processing them, they will be stored in this path. This is beneficial if you want to create a requests for use later, i.e. copy/move the requests in the requests path later.

Valid Values

- YES

Syntax: **lcc:requestsToFile** *[tab]* **[YES]**
Example: lcc:requestsToFileYES

lcc:requestMessagesReceivedMax *(optional)(one per request)*

The maximum number of messages to receive.

Syntax: **lcc:requestMessagesReceivedMax** *[tab]* **[#]**
Example: lcc:requestMessagesReceivedMax 200

lcc:requestMessagesValidMax *(optional)(one per request)*

The maximum number of valid (passed all filters) messages to parse/report.

Syntax: **lcc:requestMessagesReceivedMax** *[tab]* **[#]**
Example: lcc:requestMessagesReceivedMax 200

lcc:requestListFilePath *(mandatory/optional)(one per request)*

The path to a file containing a list of records to be used in the request. Each record should be on its own line and each column separated by tabs.

This is used with Request Types 'Create Users From List', 'Modify Users From List' and 'Delete Users From List'.
Syntax: **lcc:requestListFilePath** *[tab]* **[...]**
Example: lcc:requestListFilePath .\folder\ourList.txt

lcc:requestResponseOutputPath *(mandatory/optional)(one per request)*

For Requests that 'list' information about an account, this option will send the contents to a file.

Syntax: **lcc:requestResponseOutputPath** *[tab]* **[...]**
Example: lcc:requestResponseOutputPath .\folder\userInfo-jdoe.txt

lcc:requestListsDomain *(mandatory/optional)(one per request)*

When requesting lists (i.e. Users, Organizations, etc.) of objects, this will define what 'domain' should Google use for the query. This is beneficial if you have multiple domains hosted at Google.

Syntax: **lcc:requestListsDomain** *[tab]* **[...]**
Example: lcc:requestListsDomain my.college.edu

lcc:requestListUserMax *(optional)(one per request)*

Instructs the program to stop loading users after this amount.

Syntax: **lcc:requestListUserMax** *[tab]* **[...]**
Example: lcc:requestListUserMax 150

lcc:requestCustomerId *(mandatory/optional)(one per request)*

When listing a User(s)/Organization(s) Information, what 'customer id' should Google use for the query. This is beneficial if you have multiple domains hosted at Google and want to query all of them.
Note: the Customer Id can be found when retrieving information on users in a domain (using the Key 'lcc:requestListsDomain' and enabling the 'Customer Id' column with the 'lcc:requestListUserColumn' Key and value '7'.

Syntax: **lcc:requestCustomerId** *[tab]* **[...]**
Example: lcc:requestCustomerId E12x34y56

lcc:requestListUserColumn *(mandatory/optional)(one per request)*

When listing User(s) Information, what columns of information should be displayed. Use one of these Keys for each column desired.

The following columns are supported:

1 Id
2 Id Agreed To Terms
3 Alias
4 Change Password At Next Logon
5 Creation Time
6 Creation Time (RAW)
7 Customer Id
8 Deletion Time
9 Deletion Time (RAW)
10 Email - Address
11 Email - Custom Type
12 Email - ETag
13 Email - Is Primary
14 Email - Type
15 ETag
16 External Id - Custom Type
17 External Id - ETag
18 External Id - Type
19 External Id - Value
20 Hash Function
21 IM

22 Include In Global Address List
23 IP Whitelisted
24 Is Admin
25 Is Delegated Admin
26 Is Mailbox Setup
27 Kind
28 Last Logon Time
29 Last Logon Time (RAW)
30 Name - ETag
31 Name - Family Name
32 Name - Full Name
33 Name - Given Name
34 Non-Editable Address
35 Organization - Cost Center
36 Organization - Custom Type
37 Organization - Department
38 Organization - Description
39 Organization - Domain
40 Organization - ETag
41 Organization - Location
42 Organization - Name
43 Organization - Primary
44 Organization - Symbol
45 Organization - Title
46 Organization - Type
47 Org Unit Path
48 Password
49 Phone - Custom Type
50 Phone - ETag
51 Phone - Is Primary
52 Phone - Type
53 Phone - Value
54 Primary Email
55 Relation - Custom Type
56 Relation - ETag
57 Relation - Type

58 Relation - Value
59 Suspended
60 Suspension Reason
61 Thumbnail Photo URL

(optional) Since everything after the 2nd column is ignored, you can add the column id or notes after a column #.

Syntax: **lcc:requestListUserColumn** *[tab]* **[#]**
Example: lcc:requestListUserColumn 1
Example #2: (multiple)
lcc:requestListUserColumn 1 Id
lcc:requestListUserColumn 3 Alias

lcc:requestResponseOutputPathAppend *(mandatory)(one per request)*

When using the 'lcc:requestResponseOutputPath', this option will not create a new file, but, append to it. The only valid value is 'YES'.

Syntax: **lcc:requestResponseOutputPathAppend** *[tab]* **[YES]**
Example: lcc:requestResponseOutputPathAppend YES

lcc:requestListInColumns *(mandatory)(one per request)*

When listing a User/Organziation's Information, if this Key provided, the information will output each user/organization on a single line, with each column separated by a tab. The only valid value is 'YES'.

Syntax: **lcc:requestListInColumns** *[tab]* **[YES]**
Example: lcc:requestListInColumns YES

Request Types Keys Supported

Some keys are only detailed in the Request Logic Examples section, as they format/example is self explanatory.

Create User

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestUserId
- *(mandatory)* lcc:requestUserGivenName
- *(mandatory)* lcc:requestUserFamilyName
- *(optional)* lcc:requestUserFullName
  Note: this is a Read-Only field in Google and cannot be set at this time.
- *(mandatory)* lcc:requestUserPassword
- *(mandatory)* lcc:requestEnd

Delete User

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestUserId
- *(mandatory)* lcc:requestEnd

List User

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestUserId
- *(mandatory)* lcc:requestEnd

Create Users From List

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestListFilePath
- *(mandatory)* lcc:requestEnd

Modify Users From List

- *(mandatory)* lcc:requestBegin

- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestListFilePath
- *(mandatory)* lcc:requestEnd

Delete Users From List

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestListFilePath
- *(mandatory)* lcc:requestEnd

List User Messages

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestUserId
  (should be "me" and set the key lcc:googleAPIImpersonateId to the user you want to access)
- *(optional)* lcc:requestMessagesMax
  (maximum number of messages to retreive)
- *(optional)* lcc:requestResponseOutputPath
  (where to write a message body contents to, if not provide will output to the screen. In the path you can have "[lcc:messageId]", which will be replaced by each message id. If not provided, then all messages contents will be written to the same file.)
- *(optional)* lcc:requestMessagesReceivedMax
  Set a maximum number of message to receive.
- *(optional)* lcc:requestMessagesValidMax
  Set a maxium number of valid (those that pass all filters) messages to receive.
- *(optional)* requestMessageFilterQuery
  Set a Google Query to filter messages. https://support.google.com/mail/answer/7190?hl=en
  Example value: `from:ourbookstore@ourcollege.edu {"summer","fall"} -spring`
- *(optional)* lcc:requestMessagesIndexReportPath
  (creates a report with top level information about all messages parsed, like to/from/subject/date)
- *(mandatory)* lcc:requestEnd

List Organizations

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestCustomerId (should be "my_customer" unless you are a reseller)
- *(mandatory)* lcc:requestEnd

Create Organizations

- *(mandatory)* lcc:requestBegin
- *(optional)* lcc:requestSkip
- *(mandatory)* lcc:requestId
- *(mandatory)* lcc:requestType
- *(mandatory)* lcc:requestCustomerId (should be "my_customer" unless you are a reseller)
- *(mandatory)* lcc:requestEnd

Request Lists Columns

Note: all required columns must be supplied, but, if a column information isn't needed, i.e. if you don't want to change the 'first name', just leave that column empty.

Create Users

[lcc:createUser] *[tab]* **[User Id]** *[tab]* **[First Name]** *[tab]* **[Last Name]** *[tab]* **[Full Name]** *[tab]* **[Password]**
Example: lcc:createUser jdoe@my.college.edu John Doe Doe, John C00lP@ssw0rd

Modify Users

[lcc:modifyUser] *[tab]* **[User Id]** *[tab]* **[First Name]** *[tab]* **[Last Name]** *[tab]* **[Full Name]** *[tab]* **[Password]**
Example: lcc:modifyUser jdoe@my.college.edu John Doe Doe, John C00lP@ssw0rd

Delete Users

[lcc:deleteUser] *[tab]* **[User Id]**
Example: lcc:deleteUser jdoe@my.college.edu

Command Line Parameters

The following command line parameters can be supplied.

lcc:logicPath *(mandatory)*

Provides the Logic File to use.

Syntax: **lcc:logicPath** *[tab]* **[...]**
Example: lcc:logicPath lccGoogleAPI-logic.txt

lcc:debugMode *(optional)*

Forces the program to show additional information in some processes. At this time, the only additional information is how many Requests are in the Requests Queue when running in Back End mode.

Syntax: **lcc:debugMode** *[tab]* **[YES]**
Example: lcc:debugMode YES

lcc:backEnd *(optional)*

Runs the program in Back End mode.

Syntax: **lcc:backEnd** *[tab]* **[YES]**
Example: lcc:backEnd YES

lcc:sourceEncryptedPasswordHash *(optional)*

See this Key's definition in the Logic File section.

lcc:sourceEncryptedSaltKey *(optional)(one per Logic File)*

See this Key's definition in the Logic File section.

lcc:sourceEncryptedVIKey *(optional)(one per Logic File)*

See this Key's definition in the Logic File section.

Supplying A Logic File

Using A Logic File That Doesn't Have The Default Name

Syntax: **lccADChangesUsersPasswords.exe** *[space]* **lcc:logicPath** *[space]* **[Logic File]**
Example: lccADChangeUsersPasswords.exe lcc:logicPath ourLogicFile.txt

Logic File Examples

Back End Mode

```
---------------------------
GENERAL SETTINGS
lcc:requestsListsDelimeter   [tab]
lcc:requestResponseFromGoogleRetries   3
lcc:propercase          YES

---------------------------
ENCRYPTION
lcc:sourceEncryptedPasswordHash s0m3C00lPhr@$e+
lcc:sourceEncryptedSaltKey s0m3C00lPhr@$e
lcc:sourceEncryptedVIKey   s0m3C00lPhr@$e12
lcc:encryptArchivedRequests          YES
lcc:encryptArchivedLists      YES

---------------------------
SMTP
```

```
lcc:SMTPMailServer  mail.college.edu
lcc:SMTPNotificationsTo       someuser@college.edu
lcc:SMTPNotificationsFrom   lccGoogleAPI@college.edu
lcc:SMTPNotificationsSubject          lccGoogleAPI Notification
lcc:SMTPNotificationsSingleSendPath         .\lccNotificationsSingleSend.txt

---------------------------
PATH
lcc:abortProgramPath          .\lccGoogleAPI-abort.txt
lcc:logPath     .\lccGoogleAPI-Log
lcc:requestsPath        .\requests
lcc:responsesPath     .\responses
lcc:requestsListsPath .\requests\lists
lcc:requestsToFilePath        .\requests\toFile
lcc:indexesPath         .\requests\indexes

---------------------------
GOOGLE

lcc:googleServiceAccountClientId  1234.apps.googleusercontent.com
lcc:googleServiceAccountEmail      1234@developer.gserviceaccount.com
lcc:googlePrivateKeyPath    .\keys\ourKeyFile.p12
lcc:googlePrivateKeyPassword       notasecret
lcc:googleAPIApplicationName       lccGoogleAPI
lcc:googleAPIImpersonateId          lccgoogleapi@college.edu
lcc:googleAPIScope  AdminDirectoryUser
lcc:googleAPIScope  AdminDirectoryGroup
lcc:googleAPIScope  MailGoogleCom

---------------------------
TIMERS
lcc:multipleRequestsDelay   1000
lcc:requestResponseFromGoogleDelay      1000
lcc:heartbeatSeconds          60
lcc:requestsCheckSeconds  3
```

Command Line Mode - Encrypting A File

```
---------------------------
SETTINGS
lcc:logPath     .\lccGoogleAPI-Log
lcc:sourceEncryptedPasswordHash s0m3C00lPhr@$e+
lcc:sourceEncryptedSaltKey s0m3C00lPhr@$e
lcc:sourceEncryptedVIKey   s0m3C00lPhr@$e12
lcc:encryptFileSourcePath    .\file1.txt
lcc:encryptFileTargetPath    .\file2.txt
```

Command Line Mode - Decrypting A File

```
---------------------------
SETTINGS
lcc:logPath     .\lccGoogleAPI-Log
lcc:sourceEncryptedPasswordHash s0m3C00lPhr@$e+
lcc:sourceEncryptedSaltKey s0m3C00lPhr@$e
lcc:sourceEncryptedVIKey   s0m3C00lPhr@$e12
lcc:decryptFileSourcePath    .\file2.txt
lcc:decryptFileTargetPath    .\file3.txt
```

Request Logic Examples

Create User

```
lcc:requestBegin       YES
lcc:requestId  20140423-id1
lcc:requestType        Create User
lcc:requestUserId      jdoe@my.college.edu
lcc:requestUserGivenName John
lcc:requestUserFamilyNameDoe
lcc:requestUserFullName      Doe, John  (note: this is a Read-Only field and cannot be set at this time)
lcc:requestUserPassword      C00lP@ssw0rd
lcc:requestEnd         YES
```

Modify User

lcc:requestBegin YES
lcc:requestId 20140423-id1
lcc:requestType Modify User
lcc:requestUserId jdoe@my.college.edu
lcc:requestUserGivenName John
lcc:requestUserFamilyNameDoe
lcc:requestUserFullName Doe, John (note: this is a Read-Only field and cannot be set at this time)
lcc:requestUserPassword C00lP@ssw0rd
lcc:requestEnd YES

Delete User

lcc:requestBegin YES
lcc:requestId 20140423-id1
lcc:requestType Delete User
lcc:requestUserId jdoe@my.college.edu
lcc:requestEnd YES

List User

lcc:requestBegin YES
lcc:requestId 20140423-id1
lcc:requestType List User
lcc:requestListsDomain my.lowercolumbia.edu
lcc:requestListUserMax 150
lcc:requestUserId jdoe@my.college.edu
lcc:requestResponseOutputPath .\user-jdoe.txt
lcc:requestListUserColumn 1 Id
lcc:requestListUserColumn 2 Id Agreed To Terms
lcc:requestListUserColumn 7 Customer Id
lcc:requestListUserColumn 32 Name - Full Name
lcc:requestListUserColumn 54 Primary Email
lcc:requestEnd YES

List Users

```
lcc:requestBegin	YES
lcc:requestId	20140423-id1
lcc:requestType	List Users
lcc:requestListsDomain	my.lowercolumbia.edu
lcc:requestListInColumns	YES
lcc:requestResponseOutputPath	.\users.txt
lcc:requestListUserColumn	1	Id
lcc:requestListUserColumn	2	Id Agreed To Terms
lcc:requestListUserColumn	7	Customer Id
lcc:requestListUserColumn	32	Name - Full Name
lcc:requestListUserColumn	54	Primary Email
lcc:requestEnd	YES
```

Create Users From List

```
lcc:requestBegin	YES
lcc:requestSkip	YES
lcc:requestId	20140423-id1
lcc:requestType	Create Users From List
lcc:requestListFilePath	20140423-createUsersList.txt
lcc:requestEnd	YES
```

Modify Users From List

```
lcc:requestBegin	YES
lcc:requestSkip	YES
lcc:requestId	20140423-id1
lcc:requestType	Modify Users From List
lcc:requestListFilePath	20140423-modifyUsersList.txt
lcc:requestEnd	YES
```

Delete Users From List

```
lcc:requestBegin	YES
lcc:requestSkip	YES
lcc:requestId	20140423-id1
```

lcc:requestType Delete Users From List
lcc:requestListFilePath 20140423-deleteUsersList.txt
lcc:requestEnd YES

List Organizations

lcc:requestBegin YES
lcc:requestId 20210714-id1
lcc:requestType List Organizations
lcc:requestCustomerId my_customer
lcc:requestEnd YES

Create Organization

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Create Organizations
lcc:requestCustomerId my_customer
lcc:requestOrganizationName ourOrganization
lcc:requestOrganizationDescription Our Organization
lcc:requestOrganizationParenttOrgUnitPath /
lcc:requestOrganizationBlockInheritance false
lcc:requestEnd YES

Modify Organization (due to bug in Google's API infrastructure, we recommend not using this module)

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Modify Organizations
lcc:requestCustomerId my_customer
lcc:requestOrganizationName ourOrganization
lcc:requestOrganizationDescription Our Organization
lcc:requestEnd YES

Create Group

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Create Group

lcc:requestCustomerId my_customer
lcc:requestGroupEmail ourGroup@ourdomain.edu
lcc:requestGroupName Our Group
lcc:requestGroupDescription Our Group
lcc:requestEnd YES

Modify Group

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Modify Group
lcc:requestCustomerId my_customer
lcc:requestGroupEmail ourGroup@ourdomain.edu
lcc:requestGroupName Our Group
lcc:requestGroupDescription Our Group
lcc:requestEnd YES

Delete Group

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Delete Group
lcc:requestCustomerId my_customer
lcc:requestGroupEmail ourGroup@ourdomain.edu
lcc:requestEnd YES

Group Add Member

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Group Add Member
lcc:requestCustomerId my_customer
lcc:requestGroupEmail ourGroup@ourdomain.edu
lcc:requestUserId ouruser1@ourdomain.edu
lcc:requestUserId ouruser2@ourdomain.edu
lcc:requestEnd YES

Group Remove Member

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType Group Remove Member
lcc:requestCustomerId my_customer
lcc:requestGroupEmail ourGroup@ourdomain.edu
lcc:requestUserId ouruser1@ourdomain.edu
lcc:requestUserId ouruser2@ourdomain.edu
lcc:requestEnd YES

List Group

lcc:requestBegin YES
lcc:requestId 20210819-id1
lcc:requestType List Group
lcc:requestCustomerId my_customer
lcc:requestGroupEmail ourGroup@ourdomain.edu
lcc:requestEnd YES

Running Program Examples

Back End Mode

lccGoogleAPI.exe lcc:logicPath lccGoogleAPI-logic.txt lcc:backEnd YES

Command Line Mode (single run)

lccGoogleAPI.exe lcc:logicPath lccGoogleAPI-logic.txt

Definitions

API – Application Programmer's Interface
Encryption - http://en.wikipedia.org/wiki/Salt_(cryptography)
Password Hash – (see Encryption)
Salt Key - (see Encryption)
SMTP – Simple Mail Transfer Protocol

VI Key - (see Encryption)

Modifications

NAME	DATE	MODIFICATION
David Mielcarek	4/23/2014	Created
David Mielcarek	10/26/2016	Added folder structure for installation.
David Mielcarek	12/14/2016	Changed keys 'lcc:requestListsUsersDomain' to 'lcc:requestListsDomain' 'lcc:requestListUsersInColumns' to 'lcc:requestListInColumns' 'lcc:requestListUserColumn' to 'lcc:requestListUseColumn'
David Mielcarek	4/10/2017	Updated Request Lists Columns
David Mielcarek	8/1/2017	Added keys 'lcc:requestListUserMax', 'lcc:googleAPIScope'
David Mielcarek	8/2/2017	Added keys 'lcc:requestToFilePath', 'lcc:requestToFile'
David Mielcarek	20210714	Added Request Type "List Organizations"
David Mielcarek	20210715	Added Request Type "List User Messages"
David Mielcarek	20210719	Added filtering to List User Messsages
David Mielcarek	20210819	Added multiple new modules for organizations, groups. Updated dlls to latest versions.

		Added adhoc Logic Files under key lcc:logicAdhocPath
David Mielcarek	20210830	Added lcc:requestUserIdFromPath.

End of document