

lccADPPortal Whitepaper

Table of Contents

Description.....	1
About lccADPPortal	1
Installation.....	2
Command Line Parameters	3
Logic File.....	4
Logic File Example	9
Request File.....	9
Request File Example	17
Flag/Value Replace	19
Execution	20
Definitions	21
Modifications.....	21

Description

This document describes the technical details of the lccADPPortal.

About lccADPPortal

lccADPortal was created to provide a request/response process for Active Directory (AD) tasks.

Currently, the program includes the following abilities:

- parse Requests folder(s)
- provide Responses
- archive Requests in secured (encrypted) format
- decrypt processed Requests
- create User objects
- disable/enable User objects
- add User object to group(s)
- change account (password)
- delete account
- create file(s) for other processes
- logging

The program can check from one to multiple request(s) locations. Since each Request contains a 'request key', each location can have different capabilities. For example, one request location may only allow those requestors to create Users in a specific OU, Groups, while another allows a greater set of choices.

The program uses a Logic File that stores settings in a plain-text file using [Tab] delimiters.

The program has two possible 'Modes':

- **Back End**
Processes Requests, Responses, Logging.
- **Decrypt Request**
Used by an administrator to decrypt/view a previously processed request.
- **Front End**
A future Mode (not available at this time) to also supply the program/process for generating Requests.

Installation

1. Create the following paths:

- **Program**
A folder where the program will be placed and run from.
Example:
[P:\folder\lccADPPortal](#)

- **Requests**

Folder(s) where the program will look for Requests. This can be from one to many, depending on how many programs/processes will be providing Requests. The program will look for any files that have an extension of ".new".

Example:

`\\server\share$ \folder\Requests`

- **Requests Archive**

A folder where processed requests will be copied (the original is deleted).

Example:

`P:\folder\lccADPPortal\requestsArchive`

- **Requests Failed Archive**

A folder where failed requests will be copied (the original is deleted).

Example:

`P:\folder\lccADPPortal\requestsFailedArchive`

- **Logs**

A folder where logging will take place.

Example:

`P:\folder\lccADPPortal\logs`

2. Place the **lccADPPortal.exe** into the **Program** path.
3. Create the **Logic File** in the **Program** path.
4. Run the program according to the Command Line Parameters.

Command Line Parameters

The following Command Line Parameters can be supplied to control how the program functions.

`lcc:showLogic (optional)`

This will show what settings the program retrieves from the Logic File.

The only valid value is 'YES'.

Example:

```
lcc:showLogic "YES"
```

lcc:mode

Puts the program into a Mode. Currently, only 'BackEnd' and 'DecryptRequest' are valid.

Example:

```
lcc:mode "BackEnd"
```

lcc:debugMode (optional)

The program will provide additional information to the window and logging.

The only valid value is 'YES'.

Example:

```
lcc:debugMode "YES"
```

lcc:logicPath

The supplies the path/filename to the program for the Logic File. If one is not supplied, the program assumes a Logic File of "lccADPPortal-Logic.txt" is the same folder as the program.

Example:

```
lcc:logicPath ".\lccADPPortal-logic.txt"
```

lcc:decryptRequest

Will decrypt/display a previously processed Request. You should provide the full path, as this allows parsing processed and failed Requests. Should set the **lcc:mode** to "DecryptRequest".

Example:

```
lcc:decryptRequest ".\requests\12345.new "
```

Logic File

The Logic File provides settings to the program. The following settings are available.

lcc:logAppendYearMonthStr

Append the Year and Month to the end of the Logic File name.

Syntax:

`lcc:logAppendYearMonthStr [tab] YES`

Example:

`lcc:logAppendYearMonthStr YES`

lcc:logPath

The path/filename of the log file. The program will automatically add '.log' to the filename and optionally the Year/Month (see lcc:logAppendYearMonthStr).

Syntax:

`lcc:logPath [tab] [path]`

Example:

`lcc:logPath P:\folder\lccADPortalBackEndLog`

lcc:requestKey

Each Request process (i.e. the program/program that generates the Requests) needs to provide a Request Key. This instructs the program which settings to use.

Syntax:

`lcc:requestKey [tab] [key]`

Example:

`lcc:requestKey some cool key`

lcc:groupAuthorized

Authorizes Groups in Active Directory that can be edited by the provided Request Key. Only the Id is required by the Request, as the program will translate into the full Group Distinguished Name in Active Directory. One to many can be defined. Only those matching the Request Key will be available for each Request.

Syntax:

`lcc:groupAuthorized [tab] [key] [tab] [group id] [tab] [group OU]`

Example:

`lcc:groupAuthorized some cool key group1 OU=group1,DC=edu`

Example #2 (two groups):

`lcc:groupAuthorized some cool key group1 OU=group1,DC=edu`

`lcc:groupAuthorized some cool key group2 OU=group2,DC=edu`

lcc:createAccountOU

Sets which Organization Unit (OU) will be used when creating new User accounts. Only the Id is required by the Request, as the program will translate into the full Distinguished Name in Active Directory.

Syntax:

```
lcc:createAccountOU [tab] [key] [tab] [ou id] [tab] [ou]
```

Example:

```
lcc:createAccountOU      some cool key      create1      OU=users,DC=edu
```

lcc:searchOU

Sets which Organization Unit (OU) will be used when searching for accounts already created (to avoid process the same account twice). Only the Id is required by the Request, as the program will translate into the full Group Distinguished Name in Active Directory.

Syntax:

```
lcc:searchOU [tab] [key] [tab] [ou id] [tab] [ou]
```

Example:

```
lcc:searchOU      some cook key      search1      OU=users,DC=edu
```

lcc:requestsPath

The path where requests are being created. The program will process any file with an extension of '.new' and archive/delete the request when finished/failed.

Syntax:

```
lcc:requestsPath [tab] [key] [tab] [path]
```

Example:

```
lcc:requestsPath      some cool key      path1      \\server\share$\requests
```

lcc:responsesPath

The optional path where responses will be created. The response file will have the same filename as the request file.

Syntax:

```
lcc:responsesPath [tab] [key] [tab] [path id] [tab] [path]
```

Example:

```
lcc:responsesPath      some cool key      path1      \\server\share$\responses
```

lcc:requestsArchivePath

The path where processed requests will be copied to.

Syntax:

```
lcc:requestsArchivePath [tab] [path]
```

Example:

lcc:requestsArchivePath P:\folder\requestsArchive

lcc:requestsArchiveEncryptionOff (optional)

Instructs the program to archive requests in an unencrypted format. These file names will have the extension '.unsecured' as opposed to '.secured'. The only valid value is 'YES'. This is helpful if you want to debug request issues, i.e. you can open the '.unsecured' requests in any word processor.

Syntax:

lcc:requestsArchiveEncryptionOff [tab] [YES]

Example:

lcc:requestsArchiveEncryptionOff YES

lcc:requestsFailedArchivePath

The path where failed requests will be copied to.

Syntax:

lcc:requestsFailedArchivePath [tab] [path]

Example:

lcc:requestsFailedArchivePath P:\folder\requestsFailedArchive

lcc:pauseBetweenRequests

How long to pause between each request. Only numbers are valid. The default value if not supplied is '1'.

Syntax:

lcc:pauseBetweenRequests [tab] [number]

Example:

lcc:pauseBetweenRequests 5

lcc:ldapServer

What LDAP server will be used to process Active Directory tasks. One to many can be defined.

Syntax:

lcc:ldapServer [tab] [server]

Example:

lcc:ldapServer server1.edu

Example #2 (multiple ldap servers):

lcc:ldapServer server1.edu

lcc:ldapServer server2.edu

lcc:disableAccountException (one to many per Logic File)

Any usernames that should be ignored if an Account Disable request come in for them. i.e. high profile/sensitive accounts.

Syntax:

`lcc:disableAccountException [tab] [...username...]`

Example:

`lcc:disableAccountException jdoe`

Example #2 (multiple ldap servers):

`lcc:disableAccountException jdoe`

`lcc:disableAccountException ssmith`

lcc:deleteAccountException (one to may per Logic File)

Any usernames that should be ignored if an Account Delete request come in for them. i.e. high profile/sensitive accounts.

Syntax:

`lcc:deleteAccountException [tab] [...username...]`

Example:

`lcc:deleteAccountException jdoe`

Example #2 (multiple ldap servers):

`lcc:deleteAccountException jdoe`

`lcc:deleteAccountException ssmith`

lcc:heartbeatSeconds

How often the program will send a 'Heart Beat' message to the window/log.

Syntax:

`lcc:heartbeatSeconds [tab] [number]`

Example:

`lcc:heartbeatSeconds 600`

lcc:encryptionKey

A key used in encrypting/decrypting Requests after they have been processed.

Syntax:

`lcc:encryptionKey [tab] [key]`

Example:

`lcc:encryptionKey our coolest key`

Logic File Example

Note: extra tabs have been placed to make reading easier. There should only be one tab after each setting.

```
lcc:logAppendYearMonthStr    YES
lcc:logPath                  .\backEndLogs\lccADPortalBackEndLog
lcc:requestKey               Site 2
lcc:groupAuthorized          Site 2  Global Staff  CN=StaffGlobal,DC=ctc,DC=edu
lcc:groupAuthorized          Site 2  Adj Faculty  CN=AdjunctFaculty,CN=Listservs,DC=ctc,DC=edu
lcc:groupAuthorized          Site 2  Classified   CN=Classified,CN=Listservs,DC=ctc,DC=edu
lcc:groupAuthorized          Site 2  Exempt       CN=Exempt,CN=Listservs,DC=ctc,DC=edu
lcc:groupAuthorized          Site 2  Faculty      CN=FullTimeFaculty,CN=Listservs,DC=ctc,DC=edu
lcc:createAccountOU          Site 2  CityU        OU=NonLCCUsers,DC=ctc,DC=edu
lcc:createAccountOU          Site 2  Staff        OU=Staff,DC=ctc,DC=edu
lcc:searchOU                 Site 2  Domain - college.edu  DC=college,DC=edu
lcc:requestsPath             Site 2  \\server.edu\lccADPortalRequests$
lcc:responsesPath            Site 2  \\server.edu\lccADPortalResponses$
lcc:requestsArchivePath      Site 2  .\requestsArchive
lcc:requestsFailedArchivePath .\requestsFailedArchive
lcc:pauseBetweenRequests     5
lcc:ldapServer                dc3.college.edu
lcc:ldapServer                dc1.college.edu
lcc:heartbeatSeconds         600
lcc:encryptionKey            Some cool encryption key
```

Request File

The Request File provides settings for a single Request. The following settings are available.

General Fields Used In Most Requests

lcc:requestRoot

A 'root' id for the request. This should be the same as the filename, excluding the '.new' extension.

Syntax:

`lcc:requestRoot [tab] [root id]`

Example:

`lcc:requestRoot 20121105154748-123`

lcc:requestType

What type of request. Currently supported requests are:

- createAccount
- disableAccount
- enableAccount
- changeAccount
- deleteAccount

Syntax:

`lcc:requestType [tab] [type]`

Example:

`lcc:requestType createAccount`

lcc:requestKey

Provides a Request Key to the Back End. This key controls what settings are valid/authorized for this request. For example, what Groups can be modified.

Syntax:

`lcc:requestKey [tab] [key]`

Example:

`lcc:requestKey our cool key`

lcc:searchOU

What Active Directory (AD) Organizational Unit (OU) will be used when searching to validate the User Object does not already exist. The Request only provides the Search OU Id and not the actual OU Distinguished Name (DN). The Back End then translates the Id into the actual OU DN. Only one is valid per Request Key.

Syntax:

`lcc:searchOU [tab] [id]`

Example:

`lcc:searchOU Our Domain`

Fields Used With Create Account

lcc:createAccountMustChangePassword *(optional)*

If provided, controls the "User must change password at next logon" option on the User Account in AD. By default, this option is turned on. The only valid values are 'YES', 'NO'.

Syntax:

```
lcc:createAccountMustChangePassword [tab] [value]
```

Example:

```
lcc:createAccountMustChangePassword YES
```

lcc:createAccountCannotChangePassword *(optional)*

If provided, controls the "User cannot change password" option on the User Account in AD. By default, this option is turned off. The only valid values are 'YES', 'NO'. If the setting **lcc:createAccountMustChangePassword** is set to 'YES', then this setting is ignored.

Syntax:

```
lcc:createAccountCannotChangePassword [tab] [value]
```

Example:

```
lcc:createAccountCannotChangePassword YES
```

lcc:createAccountPasswordNeverExpires *(optional)*

If provided, controls the "Password never expires" option on the User Account in AD. By default, this option is turned off. The only valid values are 'YES', 'NO'.

Syntax:

```
lcc:createAccountPasswordNeverExpires [tab] [value]
```

Example:

```
lcc:createAccountPasswordNeverExpires YES
```

lcc:createAccountDisabled *(optional)*

If provided, controls whether the User Account is disabled. By default, this option is turned off. The only valid values are 'YES', 'NO'.

Syntax:

```
lcc:createAccountDisabled [tab] [value]
```

Example:

```
lcc:createAccountDisabled YES
```

lcc:createAccountGroup *(optional)*

What Group this new account will be added to. The Request only provides the Group Id and not the actual Group Distinguished Name (DN). The Back End then translates the Id into the actual Group DN. None to multiple can be defined.

Syntax:

`lcc:createAccountGroup [tab] [id]`

Example:

`lcc:createAccountGroup` Finance Department

Example #2 (multiple groups):

`lcc:createAccountGroup` Finance Department

`lcc:createAccountGroup` Vice Presidents

lcc:createAccountOU

What Active Directory (AD) Organizational Unit (OU) will be used when creating a new account. The Request only provides the Create Account OU Id and not the actual OU Distinguished Name (DN). The Back End then translates the Id into the actual OU DN. Only one is valid per Request Key.

Syntax:

`lcc:createAccountOU [tab] [id]`

Example:

`lcc:createAccountOU` Finance Department

lcc:createAccountCompany *(optional)*

What value will be placed into the Company attribute of the User Object..

Syntax:

`lcc:createAccountCompany [tab] [value]`

Example:

`lcc:createAccountCompany` Cool Company

lcc:createAccountCountryCode *(optional)*

What value will be placed into the Country Code attribute of the User Object..

Syntax:

`lcc:createAccountCountryCode [tab] [value]`

Example:

`lcc:createAccountCountryCode` 0

lcc:createAccountDepartment *(optional)*

What value will be placed into the Department attribute of the User Object..

Syntax:

`lcc:createAccountDepartment [tab] [value]`

Example:

`lcc:createAccountDepartment` `Finance Department`

lcc:createAccountDescription *(optional)*

What value will be placed into the Description attribute of the User Object..

Syntax:

`lcc:createAccountDescription [tab] [value]`

Example:

`lcc:createAccountDescription` `An accountant`

lcc:createDisplayName *(optional)*

What value will be placed into the Display Name attribute of the User Object..

Syntax:

`lcc:createAccountDisplayName [tab] [value]`

Example:

`lcc:createAccountDisplayName` `Doe, Jane`

lcc:createAccountDomainSuffix

What value will be appended to the Network Id for the placed into the User Principal Name attribute of the User Object..

Syntax:

`lcc:createAccountDomainSuffix [tab] [value]`

Example:

`lcc:createAccountDomainSuffix` `@college.edu`

lcc:createAccountEmployeeId *(optional)*

What value will be placed into the Employee Id attribute of the User Object..

Syntax:

`lcc:createAccountEmployeeId [tab] [value]`

Example:

`lcc:createAccountEmployeeId` `College Staff`

lcc:createAccountEmployeeType *(optional)*

What value will be placed into the Employee Type attribute of the User Object..

Syntax:

`lcc:createAccountEmployeeType [tab] [value]`

Example:

lcc:createAccountEmployeeType **Staff**

lcc:createAccountExpires *(optional)*

What value will be placed into the Expires attribute of the User Object..

Syntax:

lcc:createAccountCountryCode [tab] [value]

Example:

lcc:createAccountExpires **0**

lcc:createAccountFirstName *(optional)*

What value will be placed into the Given Name attribute of the User Object..

Syntax:

lcc:createAccountFirstName [tab] [value]

Example:

lcc:createAccountFirstName **Jane**

lcc:createAccountGroup *(optional)*

What group(s) the User Account will be added to. This can be none to many. Use the Group Id provided by the Back End (see Back End setting: **lcc:groupAuthorized**)

Syntax:

lcc:createAccountGroup [tab] [value]

Example:

lcc:createAccountGroup **Accountants**

Example #2 (multiple groups):

lcc:createAccountGroup **Accountants**

lcc:createAccountGroup **Employees**

lcc:createAccountId

What Network Id (aka cn, sAMAccountName) will be value will be used to create the User Account.

Syntax:

lcc:createAccountId [tab] [value]

Example:

lcc:createAccountId **jdoe**

lcc:createAccountLastName *(optional)*

What value will be placed into the SN attribute of the User Object..

Syntax:

`lcc:createAccountLastName [tab] [value]`

Example:

`lcc:createAccountLastName Doe`

lcc:createAccountPassword *(optional)*

What password will be set on the User Account.

Syntax:

`lcc:createAccountPassword [tab] [value]`

Example:

`lcc:createAccountPassword 12345`

lcc:createFileName *(optional)*

Note: should also use the settings - **lcc:createFileAction**, **lcc:createFileRecord**

File(s) can be created after an account has been completed. Supplying this setting will create the file at the path. This setting supports the Flag/Value Replace options. None to many can be supplied, each of these settings designates the start of a new file. Use the **lcc:createFileAction** setting to control how the file is modified, i.e. Created or Appended. To place information into the file see the **lcc:createFileRecord** setting.

Syntax:

`lcc:createFileName [tab] [value]`

Example:

`lcc:createFileName \\server\share$\folder\file.txt`

Example #2, using Flag/Value Replace:

`lcc:createFileName \\server\share$\[~lcc:createAccountId~].txt`

lcc:createFileAction *(optional)*

Note: should also use the settings - **lcc:createFileName**, **lcc:createFileRecord**

This should follow the **lcc:createFileName** setting. Controls how the file is modified, i.e. Created or Appended. The only valid values are 'Create', 'Append'.

Syntax:

`lcc:createFileAction [tab] [value]`

Example:

`lcc:createFileAction Create`

lcc:createFileRecord *(optional)*

Note: should also use the settings - **lcc:createFileAction**, **lcc:createFileAction**
This should follow the **lcc:createFileAction** setting. This setting supports the Flag/Value Replace options. None to many can be supplied, each of these settings designates another line to record into the file.

Syntax:

```
lcc:createFileRecord [tab] [value]
```

Example:

```
lcc:createFileRecord      Identity,Database
```

Example #2, using Flag/Value Replace, multiple lines:

```
lcc:createFileRecord      Identity,Database
```

```
lcc:createFileRecord      [~lcc:createAccountId~],2010Database
```

Fields Used With Disable/Enable Account

lcc:disableEnableAccountId

What Network Id (aka cn, sAMAccountName) will be value will be used to Disable/Enable an User Account.

Syntax:

```
lcc:disableEnableAccountId [tab] [value]
```

Example:

```
lcc:disableEnableAccountId      jdoe
```

lcc:disableEnableAccountDisabled *(optional)*

If provided, controls whether the User Account is disabled. By default, this option is turned off. The only valid values are 'YES', 'NO'.

Syntax:

```
lcc:disableEnableAccountDisabled [tab] [value]
```

Example:

```
lcc:disableEnableAccountDisabled      YES
```

lcc:disableEnableAccountOU

What Active Directory (AD) Organizational Unit (OU) will be used when disabling/enabling an account. The Request only provides the Disable/Enable Account OU Id and not the actual OU Distinguished Name (DN). The Back End then translates the Id into the actual OU DN. Only one is valid per Request Key.

Syntax:

```
lcc:disableEnableAccountOU [tab] [id]
```

Example:

lcc:disableEnableAccountOU

Finance Department

Request File Example

Note: extra tabs have been placed to make reading easier. There should only be one tab after each setting.

Creating An Account

lcc:requestRoot	20121005103948-70
lcc:requestType	createAccount
lcc:requestKey	Site 2
lcc:createAccountOU	Faculty And Staff
lcc:createAccountCompany	Lower Columbia College
lcc:createAccountCountryCode	0
lcc:createAccountDepartment	Staff
lcc:createAccountDescription	Staff Account - lccADPortal Test
lcc:createAccountDisplayName	testLast, testFirst
lcc:createAccountDomainSuffix	college.edu
lcc:createAccountEmployeeId	Staff
lcc:createAccountEmployeeType	Staff
lcc:createAccountExpires	0
lcc:createAccountFirstName	testFirst
lcc:createAccountGroup	Global Staff
lcc:createAccountGroup	Adjunct Faculty
lcc:createAccountId	testUser
lcc:createAccountLastName	testLast
lcc:createAccountMustChangePassword	NO
lcc:createAccountCannotChangePassword	YES
lcc:createAccountPasswordNeverExpires	YES
lcc:createAccountDisabled	NO
lcc:createAccountPassword	test
lcc:createFileName	\\server.edu\Requests\$\[~lcc:createAccountId~].new
lcc:createFileAction	Create
lcc:createFileRecord	Identity,Database
lcc:createFileRecord	[~lcc:createAccountId~],2010Staff

lcc:searchOU

college.edu

Change An Account (only passwords at this time)

lcc:requestRoot

20121005103948-70

lcc:requestType

changeAccount

lcc:requestKey

Site 2

lcc:changeAccountId

jdoe

lcc:changeAccountDomainSuffix

college.edu

lcc:changeAccountPassword

1234abcd

lcc:searchOU

college.edu

Delete An Account

lcc:requestRoot

20121005103948-70

lcc:requestType

deleteAccount

lcc:requestKey

Site 2

lcc:deleteAccountOU

Faculty And Staff

lcc:changeAccountId

jdoe

lcc:changeAccountDomainSuffix

college.edu

lcc:searchOU

college.edu

Disabling An Account

lcc:requestRoot

20121005103948-70

lcc:requestType

disableAccount

lcc:requestKey

Site 2

lcc:disableEnableAccountOU

Faculty And Staff

lcc:disableEnableAccountDisabled

YES

lcc:disableEnableAccountId

test2010

lcc:searchOU

college.edu

Enabling An Account

lcc:requestRoot

20121005103948-70

lcc:requestType	disableAccount
lcc:requestKey	Site 2
lcc:disableEnableAccountOU	Faculty And Staff
lcc:disableEnableAccountDisabled	NO
lcc:disableEnableAccountId	test2010
lcc:searchOU	college.edu

Flag/Value Replace

There are some settings that allow their values to be changed on-the-fly, i.e. when they are being processed using the current Request values. For example, you may have a Filename that will use the new User Account Id as part of its name, or the information recorded into the file.

The following Flags can be put into those supported settings values.

[~lcc:createAccountCompany~]

Will be replaced with the value supplied by [lcc:createAccountCompany](#).

[~lcc:createAccountDepartment~]

Will be replaced with the value supplied by [lcc:createAccountDepartment](#).

[~lcc:createAccountDescription~]

Will be replaced with the value supplied by [lcc:createAccountDescription](#).

[~lcc:createAccountDisplayName~]

Will be replaced with the value supplied by [lcc:createAccountDisplayName](#).

[~lcc:createAccountDomainSuffix~]

Will be replaced with the value supplied by [lcc:createAccountDomainSuffix](#).

[~lcc:createAccountEmployeeId~]

Will be replaced with the value supplied by [lcc:createAccountEmployeeId](#).

[~lcc:createAccountEmployeeType~]

Will be replaced with the value supplied by `lcc:createAccountEmployeeType`.

[~lcc:createAccountFirstName~]

Will be replaced with the value supplied by `lcc:createAccountFirstName`.

[~lcc:createAccountId~]

Will be replaced with the value supplied by `lcc:createAccountId`.

[~lcc:createAccountLastName~]

Will be replaced with the value supplied by `lcc:createAccountLastName`.

[~lcc:createAccountPassword~]

Will be replaced with the value supplied by `lcc:createAccountPassword`.

[~lcc:disableEnableAccountId~]

Will be replaced with the value supplied by `lcc:disableEnableAccountId`.

Execution

The lccADPPortal application (in Back End Mode) is a 'command line' application. This allows use in batch files, task schedulers, etc.. Since the application can change how it works, using different logic files, you can have it run multiples times, with each occurrence supplied with different settings.

While running, the program will look for a file in the same folder it is running called:

lccADPortal-abort.txt

If found, it will abort processing and delete that file. This allows you to gracefully close the program, i.e. it will finish any current tasks and close. The contents of that file do not matter.

Examples syntax

Running Program in Back End Mode

```
lccADPortal.exe lcc:showLogic YES lcc:mode "BackEnd" lcc:logicPath ".\lccADPortalBackEnd-Logic.txt"
```

Decrypting A Processed Archived Request

```
lccADPortal.exe lcc:mode "DecryptRequest" lcc:decryptRequest "[archived request path]" lcc:logicPath ".\lccADPortalBackEnd-Logic.txt"
```

Definitions

AD - Active Directory

LDAP - Lightweight Directory Access Protocol

Modifications

NAME	DATE	MODIFICATION
David Mielcarek	11/6/2012	Created
David Mielcarek	4/24/2013	Added Disable/Enable
David Mielcarek	7/18/2013	Added Key lcc:requestsArchiveEncryptionOff
David Mielcarek	20190731	Added Key lcc:disableAccountException
David Mielcarek	20190905	Added lcc:requestType 'changeAccount'
David Mielcarek	20191023	Added lcc:requestType 'deleteAccount', key lcc:deleteAccountException

End of document