lccCommander Manual

Contents

Description

This document describes the lccCommander application and how to use it.

The lccCommander application uses a text based logic file to perform commands/tasks. It was built to run in a Command window and can be embedded into batch files.

By default, lccCommander will provide brief information on results to the screen. A larger verbose level of details will be logged to a Log File.

Since any command can be passed to the 'ShellCommand', 'ExecSQL', etc.., it is assumed your account has the right permissions and your system has the right plug-ins/routes.

If using the Active Directory (AD) options, make sure the 'Interop.ActiveDS.dll' (included in the ZIP file) sits in the same folder as the program.

Use the 'Debug Levels' to increase/decrease logging information.

The program can perform that following type of tasks:

- AD queries/updates
- Change Console colors
- CMD shell commands
- Copy From Clipboard
- Copy To Clipboard
- DHCP queries/updates
- Directory listing, with extension and name filtering
- File Output/Query
- Jumping to specific locations in the logic

- Machine values (like Name, OS Version, etc.)
- Menus
- Retrieve/Set Registry Key
- Set/use variables
- SQL queries/updates
- User Input
- Variable querying with filtering

Logic File

The plain text Logic File controls what lccCommander does.

Each line:
- started with a user defined ID encapsulated with <~...~>
- lines without an encapsulated ID is considered a remark line and ignored
- is executed sequentially
- uses [tabs] to separate different parameters/values

Some parameters can have their contents changed on-the-fly by using variable names. If a parameter supports this option, it will show '*(<~!~>)*'.

When the application first runs, it will look for a Logic ID of '<~Default~>' and process that logic first.

Auto Value Replacements

- **(<~!~>)** - Any variables found will be replaced with their contents.
- **[lcc:tab]** - will be replaced with a Tab character
- **[lcc:machineName]** - will be replaced with current machine name
- **[lcc:machine64Bit]** - will be replaced with current machine Yes if 64 bit, No if not
- **[lcc:machineProcessorCount]** - will be replaced with current machine Processor Count
- **[lcc:machineUserDomain]** - will be replaced with current machine logged on User Domain
- **[lcc:machineUserName]** - will be replaced with current machine logged on User Name
- **[lcc:machineOSVersionPlatform]** - will be replaced with current machine OS Version Platform
- **[lcc:machineOSVersionServicePack]** - will be replaced with current machine OS Version Service Pack
- **[lcc:machineOSVersionBuild]** - will be replaced with current machine OS Version Build
- **[lcc:machineOSVersionMajor]** - will be replaced with current machine OS Version Major
- **[lcc:machineOSVersionMajorRevision]** - will be replaced with current machine OS Version Major Revision
- **[lcc:machineOSVersionMinor]** - will be replaced with current machine OS Version Minor

- **[lcc:machineOSVersionMinorRevision]** - will be replaced with current machine OS Version Minor Revision
- **[lcc:machineOSVersionRevision]** - will be replaced with current machine OS Version Revision
- **[lcc:machineOSVersionString]** - will be replaced with current machine OS Version String

Command Line Execution

The lccCommander program can be executed by just using its name:

lccCommander

If only the application name is used (lccCommander), it will look for a default Logic File in the same location called 'lccCommander-logic.txt'.

There are only a few Command Line parameters, beyond the application name, that can be used. Any combination of them can be used, i.e. one, or both, in any order. The value supplied to the parameter follows the parameter and a space.

Command Line Parameters

lcc:logicFile

If a different Logic File is desired, supplying the parameter 'lcc:logicFile' and the Logic File name will load that Logic File. The Logic Fi

Example:

lcc:logicFile "ourLogic.txt"

lcc:debugLevels

See the Debug Levels section for more details.

Example:

lcc:debugLevels 2

Example #2:

lcc:debugLevels 1,2,3

lcc:logFile

By Default, the Log File will be 'lccCommander-log...'. You can change this name by providing this paramter. Only provide the root/beginning of the filename. The Date (optional) and ".log" will be appended.

Example:

lcc:logFile ourLog

Logic File Commands

The following commands are recognized. Sorted alphabetically.

ADAddToGroup

Add an AD Object to an AD Group.
Columns Required: 4
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADAddToGroup *(must match this exactly)*
Column #3: Group DN *(various)* *(<~!~>)*
Column #4: User DN *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] ADAddToGroup [tab] group dn [tab] user dn
Example:
<~OurCoolId~> ADAddToGroup CN=group,DC=edu CN=user,DC=edu

ADCreateObject
Create an object in AD.
Columns Required: 6
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADCreateObject *(must match this exactly)*
Column #3: Server *(various)*(<~!~>)
Column #4: Search OU *(various)* *(<~!~>)*
Column #5: Create OU *(various)* *(<~!~>)*
Column #6: Conical Name (CN) *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] ADCreateObject [tab] (server) [tab] (search OU) [tab] (create OU) [tab] (cn)
Example:
<~OurCoolId~> ADCreateObject server.edu OU=one,DC=edu OU=two,OU=one,DC=edu jdoe

ADChangePassword
Change password on a user object.
Columns Required: 7
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADChangePassword *(must match this exactly)*
Column #3: Server *(various)*(<~!~>)
Column #4: Search OU *(various)* *(<~!~>)*
Column #5: Change OU *(various)* *(<~!~>)*
Column #6: Conical Name (CN) *(various)* *(<~!~>)*
Column #7: Password *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADChangePassword [tab] (server) [tab] (search OU) [tab] (create OU) [tab] (cn) [tab] (password)
Example:
<~OurCoolId~> ADChangePassword server.edu OU=one,DC=edu OU=two,OU=one,DC=edu jdoe newPassword

ADCreateObjectCol
Specifies an attribute/value used when creating an AD object. Should be used with and defined before 'ADCreateObject'.
Columns Required: 4
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*

Column #2: ADCreateObjectCol *(must match this exactly)*
Column #3: Column Name *(various)*
Column #4: Value *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADCreateObjectCol [tab] (name) [tab] (value)
Example:
<~OurCoolId~> ADCreateObjectCol givenName doe

ADCreateObjectColsReset

Clears all AD Object Columns. Should be used with and defined before 'ADCreateObject'.
Columns Required: 2
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADCreateObjectColReset *(must match this exactly)*
Syntax:
<~Id~> [tab] ADCreateObjectColReset
Example:
<~OurCoolId~> ADCreateObjectColReset

ADCreateObjectExpires

Set the AD Object 'expires' value. Normally set to zero (0). Should be used with and defined before 'ADCreateObject'.
Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADCreateObjectExpires *(must match this exactly)*
Column #3: 0 *(various)*
Syntax:
<~Id~> [tab] ADCreateObjectExpires [tab] number
Example:
<~OurCoolId~> ADCreateObjectExpires 0

ADCreateObjectType

Set the AD Object 'type' value. Normally set to 'user' or 'group'. Should be used with and defined before 'ADCreateObject'.
Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADCreateObjectType *(must match this exactly)*
Column #3: User *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] ADCreateObjectType [tab] type
Example:
<~OurCoolId~> ADCreateObjectType user

ADCreateObjectUserAccountControl

Set the AD Object User Account Control. A decimal value is expected. Microsoft uses it to determine the object settings, like 'Disabled', 'Password Does Not Expire', etc.. ref: http://support.microsoft.com/kb/305144. Should be used with and defined before 'ADCreateObject'.

Columns Required: 3

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADCreateObjectUserAccountControl *(must match this exactly)*
Column #3: (number) *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADCreateObjectUserAccountControl [tab] number

Example:
<~OurCoolId~> ADCreateObjectUserAccountControl 66176

ADQuery

Queries AD and places records in a variable. By default, only the first 1,000 records will be returned. To change how many are returned, use 'ADQueryPageSize' and 'ADQueryResultsSizeLimit'.

Columns Required: 8
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADQuery *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Server *(various)* *(<~!~>)*
Column #5: Search OU *(various)* *(<~!~>)*
Column #6: Object Type *(various)* *(<~!~>)*
Column #7: Search Attribute *(various)* *(<~!~>)*
Column #8: Search Value *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADQuery [tab] variable [tab] server [tab] search ou [tab] object type [tab] search attribute [tab] search value

Example:
<~OurCoolId~> ADQuery ourVars server.edu OU=one,DC=edu user cn jdoe

ADQueryPageSize

ref: http://msdn.microsoft.com/en-us/library/system.directoryservices.directorysearcher.pagesize.aspx

The maximum number of objects the server can return in a paged search. The default is zero, which means do not do a paged search.

Recommendation: If you want 'all' records, we recommend using "500" and setting lcc:resultsSizeLimit to 0.

Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ADQueryPageSize *(must match this exactly)*
Column #3: Number *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADQueryPageSize [tab] number

Example:
<~OurCoolId~> ADQueryPageSize 500

ADQueryResultsSizeLimit
lcc:resultsSizeLimit

ref: http://msdn.microsoft.com/en-us/library/system.directoryservices.directorysearcher.sizelimit.aspx

The maximum number of objects that the server returns in a search. The default value is zero, which means to use the server-determined default size limit of 1000 entries. Values above 1000 are ignored by Active Directory and will result in the default 1000 value.

Recommendation: If you want 'all' records, we recommend using "0", and setting the lcc:resultsPageSize to 500 (though any number would do).

Columns Required: 3
- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: ADQueryResultsSizeLimit *(must match this exactly)*
- Column #3: Number *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADQueryResultsSizeLimit [tab] number
Example:
<~OurCoolId~> ADQueryResultsSizeLimit 500

ADQueryColNames

What column (attribute) name/values will be returned. Should be used with and defined before 'ADQuery'.

Columns Required: 3+
- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: ADQueryColNames *(must match this exactly)*
- Column #3: Column *(various)* *(<~!~>)*
- Column #...: Column *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ADQueryColNames [tab] columns ...
Example:
<~OurCoolId~> ADQueryColNames col1 col2

ChooseFile

Set the Path/Filename of a file that will be used in later commands.

Columns Required: 3
- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: SetChooseFile *(must match this exactly)*
- Column #3: Path/Filename *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ChooseFile [tab] (Path/File)
Example:
<~OurCoolId~> ChooseFile \\server\share$\folder\file.txt

ChooseFileExtensions

Set file extensions that will be used to filter which files are used. As many as desired. Should be used with and defined before 'ChooseFile'.

Columns Required: 3+
- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: ChooseFileExtensions *(must match this exactly)*

Column #3: Extension *(various)* *(<~!~>)*
Column #...: Extension *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ChooseFileExtensions [tab] (extensions) ...

Example:
<~OurCoolId~> ChooseFileExtensions txt csv

ChooseFileFilters

Set filename filters. As many as desired. Should be used with and defined before 'ChooseFile'.

Columns Required: 3+
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ChooseFileFilters *(must match this exactly)*
Column #3: Filter *(various)* *(<~!~>)*
Column #...: Filter *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] ChooseFileFilters [tab] (filter) ...

Example:
<~OurCoolId~> ChooseFileFilters 2008May CoolFiles

ContinueAfterError

Can redirect the logic (same as using the 'RerouteLCCLogicId' key) if an error occurs. Care should be used when using this key, as almost every error can be redirected, which can easily cause an infinite loop.

Only one value is kept. Each time this key is encountered, the key value is replaced, i.e. the place where redirection goes is changed.

If you want to remove the continuation, just supply this key without a value column.

Columns Required: 2+
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ContinueAfterError *(must match this exactly)*
Column #3: Value *(various)*

Syntax:
<~Id~> [tab] ContinueAfterError [tab] (Value) ...

Example:
<~OurCoolId~> ContinueAfterError <~AnotherSection~>

Colors

Set Background and Foreground color.

See color chart at:
http://services4.lowercolumbia.edu/demo/projectlid/powerShell/colorsChart.htm

The following color values are valid for both Background and Foreground:
Black
Blue
Cyan

DarkBlue
DarkCyan
DarkGray
DarkGreen
DarkMagenta
DarkRed
DarkYellow
Gray
Green
Magenta
Red
White
Yellow

Columns Required: 4
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: Colors *(must match this exactly)*
Column #3: Color *(various)* *(<~!~>)*
Column #4: Color *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] Colors [tab] (color) [tab] (color)

Example:
<~OurCoolId~> Colors Black Yellow

CopyToClipboard

Copy the contents of a variable to the Clipboard.

Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: CopyToClipboard *(must match this exactly)*
Column #3: Variable *(various)*

Syntax:
<~Id~> [tab] CopyToClipboard [tab] (various)

Example:
<~OurCoolId~> CopyToClipboard PC-MAC

CopyFromClipboard

Copy the contents of the Clipboard to a variable.

Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: CopyFromClipboard *(must match this exactly)*
Column #3: Variable *(various)*

Syntax:
<~Id~> [tab] CopyFromClipboard [tab] (various)

Example:
<~OurCoolId~> CopyFromClipboard PC-MAC

CopyVariable

Copy the contents of a variable to another variable. If the variable being copied to already exists, it will be overwritten.

Columns Required: 4

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: CopyVariable *(must match this exactly)*
Column #3: Variable From *(various)*
Column #4: Variable To *(various)*

Syntax:

<~Id~> [tab] CopyVariable [tab] (variable from) [tab] (variable to)

Example:

<~OurCoolId~> CopyVariable PC-MAC PC-MAC-STYLE2

ExecSQL

Executes a SQL command (non query).

Columns Required: 5

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ExecSQL *(must match this exactly)*
Column #3: Server *(various)* *(<~!~>)*
Column #4: Database *(various)* *(<~!~>)*
Column #5: SQL command *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] ExecSQL [tab] server [tab] database [tab] command

Example:

<~OurCoolId~> ExecSQL server.edu db1 UPDATE table SET column1='hello world' WHERE Id='123'

FileQuery

Will query a file and return lines.

Columns Required: 4

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: FileQuery *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Path/File *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] FileQuery [tab] variable [tab] path/file

Example:

<~OurCoolId~> FileQuery NICFile \\server\share$\file.txt

FileQueryChooseCol

What column to let the user choose from when reading records from a delimited file.

Columns Required: 5

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: FileQueryChooseCol *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Path/File *(various)* *(<~!~>)*
Column #5: Column *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] FileQueryChooseCol [tab] variable [tab] path/file [tab] column

Example:

<~OurCoolId~> FileQueryChooseCol ourVar .folder\file.txt 2

FileQueryChooseColColors

What colors to show when a chosen column is found. (optional)

See Logic 'Colors' for supported color names.
Columns Required: 4
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: FileQueryChooseColColors *(must match this exactly)*
Column #3: Background Color *(various)* *(<~!~>)*
Column #4: Foreground Color *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] FileQueryChooseColColors [tab] color [tab] color
Example:
<~OurCoolId~> FileQueryChooseColColors Black Yellow

FileQueryChooseColFilters
What filters to use when stopping to allow user to choose a value. Should be used with and defined before 'FileQueryChooseCol'.
Columns Required: 3+
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: FileQueryChooseColFilters *(must match this exactly)*
Column #3: Filter *(various)* *(<~!~>)*
Column #...: Filter *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] FileQueryChooseColFilters [tab] filter ...
Example:
<~OurCoolId~> FileQueryChooseColFilters Address: IPv4

FileQueryDelimeter
What delimeterfilters to use when stopping to allow user to choose a value. Should be used with and defined before 'FileQueryChooseCol'.
Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: FileQueryDelimeter *(must match this exactly)*
Column #3: Character or [tab] *(various)*
Syntax:
<~Id~> [tab] FileQueryDelimeter [tab] delimeter
Example:
<~OurCoolId~> FileQueryDelimeter [tab]

FileQueryFilters
What filters to use when retrieving lines from the file. Should be used with and defined before 'FileQuery'.
Columns Required: 3+
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: FileQueryFilters *(must match this exactly)*
Column #3: Filter *(various)* *(<~!~>)*
Column #...: Filter *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] FileQueryFilters [tab] filter ...
Example:
<~OurCoolId~> FileQueryFilters Address: IPv4

FormatVariable

Formats the contents of a variable to meet certain requirements. You can have multiple lines of logic to perform multiple formats on the same variable.

The formats supported are:

UpperCase
: Will change any letters into uppercase letters.

LowerCase
: Will change any letters into lowercase letters.

MACStyle1
: Will place a dash '-' before every 3rd character.

MACStyle2
: Will place a colon ':' before every 3rd character.

NumbersLetters
: Will remove any characters that are not numbers or letters.

Columns Required: 4
- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: FormatVariable *(must match this exactly)*
- Column #3: Variable *(various)*
- Column #4: Format *(various)*

Syntax:
<~Id~> [tab] FormatVariable [tab] variable [tab] format

Example:
<~OurCoolId~> FormatVariable ourVar MACStyle1

Info

Display information to the console window. Each tab determines a new line of information.

Columns Required: 3+
- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: Info *(must match this exactly)*
- Column #3: Information *(various)* *(<~!~>)*
- Column #...: Information *(various)* *(<~!~>)*

Syntax:
<~Id~> [tab] Info [tab] information ...

Example:
<~OurCoolId~> Info Hello World Welcome to <~ourSchool~>

InfoNoCRLF

Same as 'Info', except without a Carriage Return/Line Feed. This will display the next 'Info' on the same line. You can change colors between this and the next to create affects like "Value = ...", where Value = is one color and ... is another color.

Syntax:
(see 'Info')

Example:
(see 'Info')

InfoCenter

Same as 'Info', except the content will be aligned centered.
Syntax:
(see 'Info')
Example:
(see 'Info')

InfoRight

Same as 'Info', except the content will be aligned right.
Syntax:
(see 'Info')
Example:
(see 'Info')

Menu

Creates a menu and places the choices value into a variable. If the variable defined is 'lccLogicId', then all logic is reprocessed with the new ID (i.e. choice from menu). Each 'Choice Value' and 'Choice Display' must be paired with each other, i.e. if you provide one, you must provide the other.
Columns Required: 7+
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: Menu *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Title of menu *(various)* *(<~!~>)*
Column #5: Prompt *(various)* *(<~!~>)*
Column #6: Choice Value *(various)* *(<~!~>)*
Column #7: Choice Display *(various)* *(<~!~>)*
Column #...: Choice Value *(various)* *(<~!~>)*
Column #...: Choice Display *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] Menu [tab] variable [tab] menu title [tab] prompt [tab] choice value [tab] choice display
Example:
<~OurCoolId~> Menu ourVar Menu Title Menu Prompt value1 Choice One value2 Choice II

MenuFromFile

Creates a menu and places the choices value into a variable. If the variable defined is 'lccLogicId', then all logic is reprocessed with the new ID (i.e. choice from menu). Choices are pulled from a [tab] delimited file, with each menu option being defined per line in the following Synax:
[Reroute Id /or/ Choice Value] [tab] [Choice Display]
example file providing values:
22222 second choice
33333 third choice
example file rerouting to other sections:
<~section2~> second choice
<~section3~> third choice

Each 'Choice Value' and 'Choice Display' must be paired with each other, i.e. if you provide one, you must provide the other.

Columns Required: 7+

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: Menu *(must match this exactly)*
- Column #3: Variable *(various)*
- Column #4: Title of menu *(various)* *(<~!~>)*
- Column #5: Prompt *(various)* *(<~!~>)*
- Column #6: File Path containing the list

Syntax:

<~Id~> [tab] Menu [tab] variable [tab] menu title [tab] prompt [tab] file path

Example:

<~OurCoolId~> Menu ourVar Menu Title Menu Prompt \\server\share$\menuList1.txt

QuerySQL

Query SQL and places results into a variable.

Columns Required: 6

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: QuerySQL *(must match this exactly)*
- Column #3: Variable *(various)*
- Column #4: Server *(various)* *(<~!~>)*
- Column #5: Database *(various)* *(<~!~>)*
- Column #6: Query Statement *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] QuerySQL [tab] variable [tab] server [tab] database [tab] query statement

Example:

<~OurCoolId~> QuerySQL ourVar server.edu db1 SELECT * FROM table1 WHERE Id='123'

QuerySQLColNames

Column names to be places into a variable. Only the number of column names provided will be returned, no matter how many are queried. The names do not need to match the queried names. Should be used with and defined before 'QuerySQLColNames'.

Columns Required: 3+

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: QuerySQLColNames *(must match this exactly)*
- Column #3: Column *(various)* *(<~!~>)*
- Column #...: Column *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] QuerySQLColNames [tab] column ...

Example:

<~OurCoolId~> QuerySQLColNames column1 column2

QuerySQLMaxResults

The maximum number of records to be returned. Should be used with and defined before ' QuerySQL'.

Columns Required: 3

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: QuerySQLMaxResults *(must match this exactly)*
Column #3: Number *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] QuerySQLMaxResults [tab] number

Example:

<~OurCoolId~> QuerySQLMaxResults 5

QueryUser

Ask the user for information. The response will be stored into a variable.

Columns Required: 3

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: QueryUser *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Prompt *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] QueryUser [tab] variable [tab] prompt

Example:

<~OurCoolId~> QueryUser usersAge What is your age

QueryUserContinueYN

Ask if the user wants to continue. If they answer 'n' (No), the program will stop processing. The application will only accept responses starting with 'y' or 'n' and will ignore anything beyond the first character.

Columns Required: 4

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: QueryUserContinueYN *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Prompt *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] QueryUserContinueYN [tab] variable [tab] prompt

Example:

<~OurCoolId~> QueryUserContinueYN usersAge What is your age

QueryUserEmpty

Same as 'QueryUser' or 'QueryUserContinueYN', except the user is allowed to respond with just pressing the ENTER key (i.e. no information).

Syntax:

(see 'QueryUserContinueYN' or 'QueryUserContinueYN')

Example:

(see 'QueryUserContinueYN' or 'QueryUserContinueYN')

QueryVariable

Queries a variable and places results into another variable.

Columns Required: 4

Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: QueryVariable *(must match this exactly)*
Column #3: Variable From *(various)*
Column #4: Variable To *(various)*

Syntax:

<~Id~> [tab] QueryVariable [tab] variable [tab] variable

Example:

<~OurCoolId~> QueryVariable variable1 variable2

QueryVariableFilters

Supplies filters to be used with ' QueryVariable '. Should be used with and defined before 'QueryVariable'.

Columns Required: 3+

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: QueryVariableFilters *(must match this exactly)*
- Column #3: Filter *(various)* *(<~!~>)*
- Column #...: Filter *(various)* *(<~!~>)*

Syntax:

<~Id~> [tab] QueryVariableFilters [tab] variable ...

Example:

<~OurCoolId~> QueryVariableFiltersvariable1 filter1 filter2

RerouteLCCLogicId

Instructs the application to start logic from the beginning, using the new lccLogicId supplied.

Columns Required: 3

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: RerouteLCCLogicId *(must match this exactly)*
- Column #3: Value *(various)*

Syntax:

<~Id~> [tab] RerouteLCCLogicId [tab] value

Example:

<~OurCoolId~> RerouteLCCLogicId <~logicSection2~>

RetrieveNICs

Retrieves NICs information for the current machine and stores into a variable. Only returns NICs that are configured for IPv4.

Columns Required: 3

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: RetrieveNICs *(must match this exactly)*
- Column #3: Variable *(various)*

Syntax:

<~Id~> [tab] RetrieveNICs [tab] variable

Example:

<~OurCoolId~> RetrieveNICs nicVar

RetrieveRegKey

Retrieve a value for a key in the registry. The response will be stored into a variable.

Columns Required: 3

- Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
- Column #2: RetrieveRegKey *(must match this exactly)*
- Column #3: Variable *(various)*
- Column #4: HKLM or HKCU
  - HKCU = HKEY Current User
  - HKLM = HKEY Local Machine

Column #5: Key Path
ex: SYSTEM\CurrentControlSet\...\
Column #6: Key Name
ex: aCoolKey

Syntax:
<~Id~> [tab] QueryUser [tab] variable [tab] prompt
Example:
<~OurCoolId~> QueryUser usersAge What is your age

SaveFileName

Creates a filename. Should be used with and defined before 'SaveToFile' or 'SaveToNewFile'.
Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: RetrieveNICs *(must match this exactly)*
Column #3: Path/File *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] SaveFileName [tab] filename
Example:
<~OurCoolId~> SaveFileName \\server\share$\folder\file.txt

SaveToFile

Same as 'SaveToNewFile', except the file will be appended to if already exists.
Columns Required: 3+
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: SaveToFile *(must match this exactly)*
Column #3: Content *(various)* *(<~!~>)*
Column #...: Content *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] SaveToFile [tab] content ...
Example:
<~OurCoolId~> SaveToFile Info for <~PC~> <~nicInfo~>

SaveToNewFile

Saves information to a new file (overwriting if one already exists). The columns can contain text and/or variables. Each column is recorded as a new line.
Syntax:
(see 'SaveToFile')
Example:
(see 'SaveToFile')

SetRegKey

Set a value for a key in the registry. The value will be stored in the key.
Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: SetRegKey *(must match this exactly)*
Column #3: Value *(various)*
Column #4: HKLM or HKCU
HKCU = HKEY Current User

HKLM = HKEY Local Machine
Column #5: Key Path
ex: SYSTEM\CurrentControlSet\...\
Column #6: Key Name
ex: aCoolKey

Syntax:
<~Id~> [tab] QueryUser [tab] variable [tab] prompt
Example:
<~OurCoolId~> QueryUser usersAge What is your age

SetVariable

Sets a value to a variable.
Columns Required: 3
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: SetVariable *(must match this exactly)*
Column #3: Variable
Column #4: Value *(various)* *(<~!~>)*, if not provided, assumed the variable will be blank.

Syntax:
<~Id~> [tab] SetVariable [tab] variable [tab] value
Example:
<~OurCoolId~> SetVariable someCoolVariable our Value
Example (setting the variable to an empty value, aka blank):
<~OurCoolId~> SetVariable our Value

ShellCommand

Executes a Shell Command and places the results into a variable.
Columns Required: 4
Column #1: <~User Defined Id~> *(variable: must enclose in <~...~>)*
Column #2: ShellCommand *(must match this exactly)*
Column #3: Variable *(various)*
Column #4: Command *(various)* *(<~!~>)*
Syntax:
<~Id~> [tab] ShellCommand [tab] variable [tab] command
Example:
<~OurCoolId~> ShellCommand netsh dhcp server \\server scope 10.10.1.0 show clients

Debug Levels

The lcc:debugLevels command line parameter increases/decreases the amount of information logged.

If this parameter is not supplied, all debugging levels are turned off (see exception to level '0', which is on by default).

You can specify one to many debug levels. If more than one specified, use a comma between the level ids.

Syntax
lcc:debugLevels [levels]
Example:
lcc:debugLevels 2
Example #2:
lcc:debugLevels 1,2,3

Debug levels

0 - errors/warnings: by default, errors/warning are logged. Supplying level '0' will turn off error/warning logging.
2 - log to screen: log records on screen
3 - skip log to file: do not log to file
4 - show logic loaded: logic records loaded
5 - function started/finished: when functions are started/finished
6 - show Query File records: QueryFile records
7 - set colors to: changing colors
8 - output to console: what is outputted to console
9 - NIC info: information gathered by NIC query
10 - processing logic: logic records as they are processed
11 - Save File: setting save file
12 - Query SQL: querying SQL
13 - AD Create Object: creating object in AD
14 - AD Query: querying AD
15 - Choose File: choosing a file
16 - File Query: querying a file
17 - Query Variable: querying a variable
18 - User input: receiving user input
19 - Copy Variable: copying a variable
20 - Copy to clipboard: copying to clipboard
21 - Format Variable: formatting a variable
22 - Shell Command: executing shell command
23 - Exec SQL: executing SQL command
24 - AD Add To Group: adding object to group
25 - Save To File: saving to file
26 - Replace Tags: replacing tags
27 - Format Value: checking validity of value contents (i.e. numbers only)
28 - AD Change Password: changing password on AD User
29 - Retrieve Registry Key: retrieving registry key/value

Script Examples

Due to the length of logic rows, each row will be separated in the examples below by '.....', though these lines would not be in the logic.

These scripts have generic information that should be modified before use/testing.

#1: Create a menu using Black/White colors. Echo the results to the console in Black/Yellow colors.

```
<~Default~>    Colors  Black   White
.............................
<~Default~>    Menu    echoMyVar      What would you like shown Earth      Name Of The Planet
       Blue    Color Of The Sky
.............................
<~Default~>    Colors  Black   Yellow
.............................
<~Default~>    Info    echoMyVar
```

#2: Display banner, create a menu, then reroute logic depending on user choice.

```
<~Default~>    SetVariable    menuMode       Original Mode
.............................
<~Default~>    Colors  Blue    White
.............................
<~Default~>    InfoCenter     ================================================  Welcome to the
lccCommander program.  Copyright (c) Lower Columbia College
       ================================================
.............................
<~Default~>    Colors  Black   White
.............................
<~Default~>    Menu    lccLogicId     <~menuMode~> Of Operation     Choice: <~GatherInformation~>
       Gather NIC Information <~Record802AuthInformation~>  Record 802 Auth Information
       <~ExitProgram~>        Exit Program
.............................
<~GatherInformation~>  Colors  DarkGray       White
<~GatherInformation~>  InfoRight      Gathering Information For 802.1X
.............................
<~Record802AuthInformation~>  Colors  DarkGray       White
<~Record802AuthInformation~>  InfoRight      Recording Information For 802.1X
.............................
<~ExitProgram~>        Info    Exiting Program, Due To User Request
```

#3: Query user for key and information, save information to file with a dynamic filename using key and save with preset header/footer.

```
<~Default~>    QueryUser      fileKey Key
.............................
<~Default~>    QueryUser      userInfo       Information you would like saved to a file
.............................
<~Default~>    SaveFileName   coolFile-<~Key~>.txt
.............................
<~Default~>    SaveToNewFile  our header line       <~userInfo~>
.............................
<~Default~>    SaveTo  our footer
```

#4: Query NICs and save to file with computer TAG #.

```
<~Default~>    QueryUser      TAG     TAG NUMBER:
.............................
<~Default~>    RetrieveNICs   NICsInformation
.............................
<~Default~>    Colors  DarkGray       White
.............................
<~Default~>    Info    ...Information Gathered.
.............................
<~Default~>    SaveFileName   nicInfo-<~TAG~>.txt
.............................
<~Default~>    Colors  DarkGray       White
.............................
```

```
<~Default~>    Info    Saving To File nicInfo-<~TAG~>.txt
..............................
<~Default~>    SaveToNewFile  lccCommander NIC Information for <~TAG~>     NICsInformation
```

#5: Perform SQL Query and show only the first five results.

```
<~Default~>    Colors  DarkGray       White
..............................
<~Default~>    Info    Finding next five (5) available IPs
..............................
<~Default~>    QuerySQLColNames       Address
..............................
<~Default~>    QuerySQLMaxResults     5
..............................
<~Default~>    Colors  Black   Yellow
..............................
<~Default~>    QuerySQL       SQLNextOpenIP  sqlserver.edu  ipsDatabase    SELECT
RTRIM(Address),Id,RTRIM(Subnet),RTRIM(MAC),RTRIM(VLAN),RTRIM(Username),RTRIM(ComputerName),RTRIM(
tagNumber),RTRIM(Location) FROM ipsTable WHERE VLAN LIKE '<~IPListNextFiveVLAN~>%' AND
(tagNumber='' OR tagNumber IS NULL) AND (computerName='' OR computerName IS NULL) AND (MAC='' OR
MAC IS NULL) AND (Username='' OR Username IS NULL) ORDER BY Address;
..............................
<~Default~>    Colors  DarkGray       White
..............................
<~Default~>    Info    SQL Query Results
..............................
<~Default~>    Colors  Black   Yellow
..............................
<~Default~>    Info    <~SQLNextOpenIP~>
```

#6: Perform SQL command with preset variables.

```
<~Default~>    SetVariable    IPList-MAC     121212121212
..............................
<~Default~>    SetVariable    IPList-TAGNumber       AB-123
..............................
<~Default~>    SetVariable    IPList-Location        Admin, Room 123
..............................
<~Default~>    SetVariable    IPList-Username        jdoe
..............................
<~Default~>    SetVariable    IPList-Notes   A very cool computer/user combination
..............................
<~Default~>    ExecSQL server.edu     ipsDatabase    UPDATE ipsTable SET MAC='<~IPList-
MAC~>',tagNumber='<~IPList-TAGNumber~>',Location='<~IPList-Location~>',Username='<~IPList-
Username~>',Notes='<~IPList-Notes~>' WHERE Address='<~IP~>';
```

#7: Add AD Object to AD Group

```
<~Default~>    Colors  DarkGray       White
..............................
<~Default~>    InfoRight      Recording Machine To Security Group: <~AD-MAC~>
..............................
<~Default~>    ADAddToGroup   CN=<~RadiusGroup~>,OU=RadiusGroups,DC=edu    CN=<~AD-
MAC~>,OU=<~RadiusGroupSubOU~>,DC=edu
```

#8: Query AD Objects

```
<~Default~>    SetVariable    MAC     123.123.123.123
..............................
<~Default~>    Colors  DarkGray       White
..............................
```

```
<~Default~>    CopyVariable   MAC     AD-MAC
.............................
<~Default~>    InfoRight      Checking MAC <~AD-MAC~> Against AD
.............................
<~Default~>    Colors  DarkGray       White
.............................
<~Default~>    FormatVariable NumbersLetters AD-MAC
.............................
<~Default~>    FormatVariable LowerCase      AD-MAC
.............................
<~Default~>    ADQueryColNames        distinguishedName      cn
.............................
<~Default~>    Colors  Black   Yellow
.............................
<~Default~>    ADQuery ADCheckMAC     server.edu     OU=RadiusGroups,DC=edu User    cn      <~AD-
MAC~>
```

#9: Query File with filters

```
<~Default~>    Colors  Yellow  Black
<~Default~>    SetVariable    NICFile .\folder\file.txt
<~Default~>    FileQueryFilters       NIC-Name       NIC-MAC
<~Default~>    FileQuery      NICFile2       <~NICFile~>
<~Default~>    Info    NICFile2
<~Default~>    Colors  Green   Black
<~Default~>    Info    <~NICFile2~>
```

#10: Continue After Error

```
<~Default~>    ContinueAfterError     <~OurErrorSection~>
.............................
<~Default~>    Colors  DarkGray       White
.............................
<~Default~>    InfoRight      Purposefully creating error
.............................
<~Default~>    ADAddToGroup   CN=InvalidCN,OU=InvalidOU,DC=xyz      CN=Invalid,OU=Invalid,DC=xzz

<~OurErrorSection~>    InfoRight      Our created error happend, but, will continue.
.............................
<~OurErrorSection~>    InfoRight      Continuing
.............................
```

#11: Change Password

```
<~Default~>    SetVariable    DomainController       server.edu
<~Default~>    QueryUser      NetworkId      Network Id To Change:
<~Default~>    QueryUser      Password       Change Password To:
<~Default~>    ADChangePassword       <~DomainController~>   DC=college,DC=edu
       OU=Staff,DC=college,DC=edu    <~NetworkId~>  <~Password~>
<~Default~>    Info    Change Password Done
.............................
```

#12: Retrieve Registry Key/Value for Computer Description, Change It, Retrieve Again, Display

```
<~Default~>    RetrieveRegKey lccMachineDescription  HKLM
       SYSTEM\CurrentControlSet\Services\lanmanserver\parameters   srvcomment
<~Default~>    Info    Machine Description [<~lccMachineDescription~>]
<~Default~>    SetRegKey      Workstation for David Mielcarek       HKLM
       SYSTEM\CurrentControlSet\Services\lanmanserver\parameters   srvcomment
```

```
<~Default~>    RetrieveRegKey lccMachineDescription2 HKLM
       SYSTEM\CurrentControlSet\Services\lanmanserver\parameters   srvcomment
<~Default~>    Info    Machine Description [<~lccMachineDescription2~>]
```

.....

Definitions

AD - Active Directory
CMD - Windows Command Line window
DHCP - Dynamic Host Configuration Protocol
NIC - Network Interface Card
OS - Operating System
SQL - Structured Query Language

Modifications

NAME	DATE	MODIFICATION
David Mielcarek	8/22/2012	Created
David Mielcarek	8/29/2012	Updated most parameters to allow replacing of content from variables.
David Mielcarek	10/12/2012	Updated manual format and added 'ContinueAfterError' key.
David Mielcarek	12/07/2012	Added ADChangePassword
David Mielcarek	2/17/2016	Added [lcc:tab], multiple machine environment values, ex: [lcc:computerName] replacement option, CopyFromClipBoard
David Mielcarek	2/22/2016	Added RetrieveRegKey and SetRegKey.
David Mielcarek	2/22/2016	Added MenuFromFile

End of document