

# lccAuditPasswordsLib

## Manual

### Contents

Description.....	1
lccAuditPasswordsLib .....	1
Embedding The Library In Other Programs.....	2
Main Functions/Variables Available In The Library .....	2
Code Example Inside Another Program .....	3
Logic File.....	4
lcc:logPath ( <i>optional</i> )( <i>none to one</i> ) .....	4
lcc:debugLevel ( <i>optional</i> )( <i>none to many</i> ) .....	4
lcc:auditPassword ( <i>optional</i> )( <i>none to many</i> ) .....	5
lcc:passwordMinimumRulesMatched ( <i>mandatory</i> )( <i>one per Logic File</i> ).....	5
lcc:passwordMinimumLength ( <i>optional</i> )( <i>one per Logic File</i> ).....	5
lcc:passwordMaximumLength ( <i>optional</i> )( <i>one per Logic File</i> ) .....	5
lcc:passwordGoogleMinimumRating ( <i>optional</i> )( <i>one per Logic File</i> ) .....	5
lcc:passwordCharacterSet ( <i>optional</i> )( <i>none to many</i> ).....	6
lcc:googleRatingTitle ( <i>optional</i> )( <i>none to many</i> ) .....	6
lcc:googleRatingURL ( <i>optional</i> )( <i>one per Logic File</i> ) .....	6
Definitions .....	6
Modifications .....	6

---

## Description

This document describes how to use the lccAuditPasswordsLib.

### **lccAuditPasswordsLib**

is a library (dll) created to be used within other programs, i.e. Web, GUI, Command Line.

The library will audit passwords against a set of dynamic requirements and Google Rating.

The library receives its settings from a Logic File to provide the requirements/settings to audit passwords, retrieve Google Ratings.

---

## Embedding The Library In Other Programs

Note: code examples shown in C# (using Visual Studio 2013).

To embed the library in another program:

- copy the lccAuditPasswordsLib.dll to your programs folder
- add a reference to the lccAuditPasswordsLib.dll in your project
- instantiate the library class
- example:

```
static lccAuditPasswordsLib.lccAuditPasswordsLib lccAPLLib = new  
lccAuditPasswordsLib.lccAuditPasswordsLib();
```

---

## Main Functions/Variables Available In The Library

### lccFReturnSettings

lccFReturnSettings().lccBAAbortProgram  
lccFReturnSettings().lccSLogicPath  
lccFReturnSettings().lccALAuditPasswords  
lccFReturnSettings().lccALResponseDetails  
lccFReturnSettings().lccBResponse

### lccFBuildPasswordRequirements

### lccFProcessPasswords()

### lccFLogInfo

### **lccFReturnSettings** (*class*)

Gets/Sets variables used by the library.

### **lccFReturnSettings().lccBAAbortProgram** (*boolean*)

Gets/Sets a boolean (true/false) variable. If set to true, the library will cease further tasks. The library also sets this to true if something goes wrong.

### **lccFReturnSettings().lccSLogicPath** (*string*)

Gets/Sets the Logic File path, i.e. logic used by the Library to build the requirements/process tasks.

### **lccFReturnSettings().lccALAuditPasswords** (*List<string>, aka array*)

Gets/Sets the password(s) to be audited.

### **lccFReturnSettings().lccALResponseDetails** (*List<string>, aka array*)

Gets the response details provided by the library after processing tasks.

### **lccFReturnSettings().lccBResponse** (*boolean*)

Gets the validity of a password audit. True=valid, False=invalid (failed requirements)

### **lccFBuildPasswordRequirements** (*function*)

Loads the requirements from the Logic File and builds the requirements for auditing passwords.

### **lccFProcessPasswords** (function)

Audits the passwords against the requirements and builds the lccALResponseDetails and lccBResponse values.

### **lccFLogInfo** (function)

Logs information (if a lcc:logicPath provided in the Logic File)

---

## Code Example Inside Another Program

This example using the library functions/variables to perform audits on passwords provided to a command line program (in this case, lccAuditPasswords). This code also provides the response(s) and results, i.e. met requirements or not.

Note: this example is C# used in Visual Studio 2013.

Note #2: ellipses (...) are used where other code was used, but, not need for this example.

```
static lccAuditPasswordsLib.lccAuditPasswordsLib lccAPLLib = new
lccAuditPasswordsLib.lccAuditPasswordsLib();
static void Main(string[] lccALArgs)
{
    try
    {
        if (lccAPLLib.lccFReturnSettings().lccBAbortProgram == false)
        {
            if (lccAPLLib.lccFReturnSettings().lccSLogicPath.Length == 0)
            {
                lccAPLLib.lccFReturnSettings().lccSLogicPath = @".\lccAuditPasswords-logic.txt";
                lccAPLLib.lccFBuildPasswordRequirements();
                if (lccAPLLib.lccFReturnSettings().lccALAuditPasswords.Count == 0)
                {
                    if (lccAPLLib.lccFReturnSettings().lccALResponseDetails.Count > 0)
                    {
                        foreach (string lccSResponseLine in
lccAPLLib.lccFReturnSettings().lccALResponseDetails)
                        {
                            Console.WriteLine(lccSResponseLine);
                        }
                    }
                }
            }
        }
        if (lccAPLLib.lccFReturnSettings().lccALAuditPasswords.Count == 0)
        {
            Console.WriteLine("No passwords provided to audit.");
            lccAPLLib.lccFReturnSettings().lccBAbortProgram = true;
        }
        if (lccAPLLib.lccFReturnSettings().lccBAbortProgram == false)
        {
            if (lccAPLLib.lccFReturnSettings().lccBAbortProgram == false)
            {
                lccAPLLib.lccFProcessPasswords();
            }
        }
    }
}
```

```

    if (lccAPLLib.lccFReturnSettings().lccALResponseDetails.Count > 0)
    {
        foreach (string lccSResponseLine in
lccAPLLib.lccFReturnSettings().lccALResponseDetails)
        {
            Console.WriteLine(lccSResponseLine);
        }
    }
    if (lccAPLLib.lccFReturnSettings().lccALResponseDetails.Count == 0)
    {
        if (lccAPLLib.lccFReturnSettings().lccBResponse == true)
        {
            Console.WriteLine("Congratulations, your password met all requirements");
        }
        if (lccAPLLib.lccFReturnSettings().lccBResponse == false)
        {
            Console.WriteLine("Sorry, your password did not meet all requirements");
        }
    }
}
catch (Exception lccException)
{
    lccAPLLib.lccFLogInfo(0, 0, "[main] ERROR: " + lccException.Message);
}
}

```

---

## Logic File

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.

Note: on Rules, the Optional/Mandatory column lets you specify more rules than are required. For example, you may have a requirement that a password match 3 out of 4 rules. If those 4 rules are set as Optional, but, you set the 'must match # rules' to '3', the password can pass any of the four Optional rules to be valid.

The Logic File uses the syntax:

**Syntax:** **[Key]** *[tab]* **[Value]**

**Example:** **lcc:key**      **value**

**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 Logic Loaded. Will display settings retrieved from the Logic File.
- 2 Show Process Passwords process.
- 3 Show Rules Passed.

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

Example: **lcc:debugLevel** 3

Example #2 (multiple):

**lcc:debugLevel** 1  
**lcc:debugLevel** 3

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

You can supply passwords through your program that uses this library, and/or provide password directly through the Logic File.

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

Example: **lcc:auditPassword** myC00lP@ssword

Example #2 (multiple):

**lcc:auditPassword** myC00lP@ssword  
**lcc:auditPassword** anotherpa\$\$word

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

How many Rules must the password match to be considered valid.

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

Example: **lcc:passwordMinimumRulesMatched** 6

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

How long a password must be to pass this rule.

Syntax: **lcc:passwordMinimumLength** [tab] [#] [tab] [Optional/Mandatory]

Example: **lcc:passwordMinimumLength** 10

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

How short a password must be to pass this rule.

Syntax: **lcc:passwordMaximumLength** [tab] [#] [tab] [Optional/Mandatory]

Example: **lcc:passwordMaximumLength** 10

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

What Rating # the password must make in Google Ratings to pass this rule.

Must also define the Key **lcc:googleRatingURL**, and recommend defining Key **lcc:googleRatingTitle** for each rating possible.

ref: <https://www.google.com/accounts/RatePassword?Passwd=1234>

Syntax: **lcc:passwordGoogleMinimumRating** [tab] [#] [tab] [Optional/Mandatory]

Example: **lcc:passwordGoogleMinimumRating** 3

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

A set of characters that the password must match the quantity specified to pass this rule. The columns are:

- **#**: How many characters of the set must match.
- **Optional/Mandatory**: Is the rule mandatory.
- **Title**: What Title will show in the report when the set fails.
- **Set**: What characters are in the set.

**Syntax:** **lcc:passwordCharacterSet** [tab]  **[# ]** [tab]  **[Optional/Mandatory]** [tab]  **[Title]** [tab]  **[..set..]**

**Ex:** **lcc:passwordCharacterSet**       **2**       **Optional**       **abcdefghijklmnopqrstuvwxyz**

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

What Google Ratings mean and what to display in the report. For example, if Google Rating gives your password a '3', what do you want that called, i.e. 'Good'.

ref: <https://www.google.com/accounts/RatePassword?Passwd=1234>

**Syntax:** **lcc:googleRatingTitle** [tab]  **[# ]** [tab]  **[Title]**

**Example:** **lcc:googleRatingTitle**               **3**       **Good**

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

What is the URL to do a Google Rating call. Put the flag "[lcc:checkPassword]" anywhere in the URL where you want the password being audited to be placed.

**Syntax:** **lcc:googleRatingURL** [tab]  **[URL]**

**Example:**

**lcc:googleRatingURL**      [https://www.google.com/accounts/RatePassword?Passwd=\[lcc:checkPassword\]](https://www.google.com/accounts/RatePassword?Passwd=[lcc:checkPassword])

---

## Definitions

**DLL** - Dynamic Link Library

---

## Modifications

NAME	DATE	MODIFICATION
David Mielcarek	4/3/2014	Created

---

End of document