lccRequestMaker
White Paper

Contents

Description

This document describes how to install/configure the lccRequestMaker program.

Note: the lccRequestMaker program relies on lccRequestMakerBackEnd. Please see that white paper as well.

lccRequestMaker

was created to allow end users to submit requests for processing. This allows users with no elevated rights/access, to request/have processes ran by a designated IT elevated user/processes.

Please see the PDF lccRequestMaker-flow.pdf for a visual of how the process works.

Installation

- create a Logic File for the end user(s)
- place the Logic File in an accessible location for the user (recommend read-only rights)
- place the lccRequestMaker program in an accessible location for the user
- create a short pointing to the program, with the follow parameters:
    - lcc:logicPath
    - ...path...
    - Example shortcut target:
    - \\server\share\$\lccRequestMaker.exe lcc:logicPath \\server\share$\lccRequestMaker-logic.txt

Run the program with the following syntax:
Syntax: **lccRequestMaker.exe lcc:logicPath [...path...]**
Example: lccRequestMaker.exe lcc:logicPath lccRequestMaker-logic.txt

Logic File Description/Syntax

A 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.

The Logic File uses the syntax.
Syntax: **[Key]** *[tab]* **[Value]** … *[tab]* **[Value]**
Example: lcc:key value

Any extra tabs in a line after the expected ones are considered remarks and will be ignored. This is a nice way to document specific Key settings (see Log Levels in the Logic File example(s) for reference). Also, if you place a tab before a line, that will essentially make it a remark and will be ignored, which makes using/not using logic without removing quicker.

Any line not starting with an expected key is ignored, which makes placing remarks/formatting the logic easy.

Logic File

The Logic File will contain one to many authorized Requests. Each authorized Request started with the key "lcc:requestSetId".

lcc:requestSetId *(mandatory, one to many per Logic File)*

Starts and ids an authoried Request Id. The 'id' can be anything, but must match authorized Request Ids on the Back End.

Syntax: lcc:requestSetId *[tab]* [..id..]
Example: lcc:requestSetId everfiFoundryStudents

lcc:requestSkip *(optional, one to many per Logic File, one per Request Id)*

If provided, and 'YES', will skip using this Request Id set.

Valid Values:
- **YES**

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

lcc:requestTitle *(mandatory, one to many per Logic File, one per Request Id)*

Provides the user friendly title to display in the Available Requests.

Syntax: lcc:requestTitle *[tab]* [...title...]
Example: lcc:requestTitle EverFi Foundry Students

lcc:requestPath *(mandatory, one to many per Logic File, one per Request Id)*

What path request file(s) can be created at.

Syntax: lcc:requestPath *[tab]* [...path...]
Example: lcc:requestPath \\server\lccRequestMaker -requests$

lcc:requestFilename *(mandatory, one to many per Logic File, one per Request Id)*

What request filename to create. The filename is looked for by the backend. Once found, the back end looks at the username who created the file for authorization. The file needs no contents.

Syntax: lcc:requestFilename *[tab]* [...filename...]
Example: lcc:requestFilename lccSQLDataExportRequest-everFiFoundry-students.txt

lcc:requestSubmittedResponse *(mandatory, one to many per Logic File, one per Request Id)*

What message to display to the user after the request has been submitted.

Syntax: lcc:requestSubmittedResponse *[tab]* [...response...]
Example: lcc:requestSubmittedResponse Your EverFi Foundry Students report should be ready within a minute.

lcc:requestSubmittedResponsePath *(optional, one to many per Logic File, one per Request Id)*

If provided, after the response message has been clicked and the user clicks OK, the program will open this path. This is beneficial if you are provided data/file(s) to the end user.

If the path starts with 'http', their web browser will open the location. If not, their Windows Explorer will (i.e. network location).

Syntax: lcc:requestSubmittedResponsePath *[tab]* [...path...]
Example: lcc:requestSubmittedResponsePath \\server\lccRequestMaker-exports-students$

Logic File Examples

A Single Request Id

```
lcc:requestSetId    everfiFoundryStudents
      lcc:requestSkip     YES
lcc:requestTitle    EverFi Foundry Students
lcc:requestPath     \\server\lccRequestMaker-requests$
lcc:requestFilename lccSQLDataExportRequest-everFiFoundry-students.txt
lcc:requestSubmittedResponse     Your EverFi Foundry Students report should be ready within a minute.
lcc:requestSubmittedResponsePath \\server\lccRequestMaker-exports-students$
```

Multiple Requests

```
lcc:requestSetId    everfiFoundryStudents
       lcc:requestSkip     YES
lcc:requestTitle    EverFi Foundry Students
lcc:requestPath     \\server\lccRequestMaker -requests$
lcc:requestFilename lccSQLDataExportRequest-everFiFoundry-students.txt
lcc:requestSubmittedResponse     Your EverFi Foundry Students report should be ready within a minute.
lcc:requestSubmittedResponsePath \\server\lccRequestMaker-exports-students$

lcc:requestSetId    everfiFoundryEmployees
       lcc:requestSkip     YES
lcc:requestTitle    EverFi Foundry Employees
lcc:requestPath     \\server\lccRequestMaker -requests$
lcc:requestFilename lccSQLDataExportRequest-everFiFoundry-employees.txt
lcc:requestSubmittedResponse     Your EverFi Foundry Employees report should be ready within a minute.
lcc:requestSubmittedResponsePath \\server\lccRequestMaker-exports-students$
```

Definitions

Back End - where the lccRequestMakerBackEnd processes requests
IT - Information Technology

Modifications

NAME	DATE	MODIFICATION
David Mielcarek	7/27/2020	Created

End of document