Maker Nexus Card Writer API

Version 1.2; by: Bob Glicksman & Jim Schrempp; 10/05/19

Overview.

The Maker Nexus *Card Writer* application will consist of two parts:

- GUI -- the GUI will be an AI2 app that provides a user interface for a Maker Nexus administrator.
- Particle Firmware -- firmware for an Argon or Xenon that communicates with an RFID card and with EZFacility in order to implement the function selected by the administrator via the GUI.

The detailed requirements are contained in a separate Concept and Requirements document:

https://docs.google.com/document/d/13DphSTjkPGKusOAcMALxCAwljKkrsXL4rh26epaUW9I/edit#heading=h.jfvwj4z7t60w

This document describes the API between the GUI and the Firmware. The API uses Particle.function() to expose functions in the Firmware to the Particle Cloud, Particle.variable() to expose variables to the Particle Cloud, and the Particle REST interface to communicate GUI commands and status over the Internet to the Particle Cloud.

The Card Writer consists of three functions:

- Make a factory fresh card into an MN formatted card for a specific member.
- Reset an MN formatted card to factory fresh condition
- Identify an unknown card

Format a Factory Fresh Card for a Member.

The GUI will have a text box for the administrator to manually type in the Membership Number for the member for whom a card is to be produced. The GUI will have a button that activates the firmware to query EZFacility by membership number and return and display membership information from EZFacility for verification by the administrator. Once the administrator has verified that the member is the correct one, another button will become available to command “burning” the RFID card for that member.

The function prototype for asking the firmware to look up the member by number in EZFacility will be as follows:

int *queryMember*(String *memberNumber*);

Where:

memberNumber is a String representation of the number typed into the GUI by the administrator.

The return value is an int with the following encoding:

0 = query accepted
1 = query is underway
2 = query is done and results are in the cloud variable (including any error report)
3 = query was already underway, this query is rejected
4 = memberNumber was not a number or was 0
5 = other error, see LCD panel on device

The expected success pattern is to call queryMember and get a 0 return code. Then poll queryMember with the same memberNumber every one second. If a poll returns 1, poll again in one second. If a poll return 2, then access the cloud variable. At some point the application will stop polling, assuming the firmware has failed to respond correctly.

Successful query will load the JSON representation of member data from EZ Facility into the cloud variable:

String *queryMemberResult*;

Where *queryMemberResult* is JSON formatted name:value pairs. The GUI will present each JSON name as a literal and each JSON value next to the variable name. The returned name:value pairs shall be:

ErrorCode: 0
ErrorMessage: OK
Name : Bob Glicksman
ClientID : 12345678
Status : Active
Checkin: Allowed (or another message that would be displayed if a checkin is attempted)

Where ErrorCode is

0 = member found and membership data is in the JSON.
1 = member not found in EZFacility
2 = more than one member found in EZfacility (this should be impossible)

3 = other error.

ErrorMessage is a short string in English, suitable for display to a user.

Note that other information may be provided in the JSON response as deemed appropriate from the information that is in EZ Facility. The purpose of this information is for the administrator to verify that the card is being made for the correct member (e.g. that they did not mistype the membership number). The application should display the required fields to the user (as specified above). Additional information in the JSON response may be displayed if it is decided that this additional information is helpful to the administrator and does not unduly compromise member privacy.

At this point the LCD on the device will display “Selected User” and the user’s name.

After return of correct membership information, the GUI will present a “burn card” button to the administrator. Tapping this button will result in the following Cloud function call:

int *burnCard*(String *clientID*);

Where:

clientID is a String representation of the clientID that was returned in the membership information JSON response above

The return value is an int with the following encoding:

0 = function called successfully; follow instructions on the LCD
1 = failed to contact card reader/writer

.

The user will be prompted on the LCD panel and/or LEDs when to remove the card and what the burn status is (good or fail).

Note that prompts for the administrator to place the card on the card writer and to subsequently remove the card will be performed by the Firmware using the hardware LCD (rather than the GUI). The LCD is right where the card is to be placed and is therefore the logical place for such user prompting.

If the burnCard function is not activated within a timeout period after the memberQuery call, the device firmware will clear itself and return to a ready state.

Reset a MN Card to Factory Fresh.

A member resigns and turns in their card. The cards are not expensive, but we might as well recycle them. The easiest way to recycle a card is to make it factory fresh. Otherwise, we rely on the MN format staying the same -- i.e. the secret keys not changing. If we change keys and forget about recycled cards, they become useless. Factory fresh always has known default keys. The function will only reset cards that are valid with the most recent previous encryption write key. Cards constructed with older encryption write keys cannot be reused.

The GUI will have a button to select this function. The button will result in the following Cloud function call:

int *resetCard*();

The return value is an int with the following encoding:

0 = function called; follow instructions on the LCD panel
1 = failure to contact the card reader/writer hardware
2 = device is busy with other operation

Note that prompts for the administrator to place the card on the card writer and to subsequently remove the card will be performed by the Firmware using the hardware LCD (rather than the GUI). The LCD is right where the card is to be placed and is therefore the logical place for such user prompting.

Test A Card.

This function is used for several things.

1. If a card is found with out any markings, this function can be used to read the card and determine who it belongs to, if it is bad, or some non-MN format.
2. Once a card has been burned, this function can be used to verify that the card has been correctly burned.
3. If a member has failed to checkin (red light) this function can be used by a manager in the office to discover why a member's checkin failed and discuss the issue privately with the member.
4. If a member corrects the problem that caused the failed checkin (e.g. paid their outstanding balance) this function can be used to verify that the card will successfully work at a checkin station now.
5. If a member is incorrectly denied checkin (red light) the manager can use this function to test the card as they attempt to correct the problem.

The GUI will have a button to select this function. The button will result in the following Cloud function call:

int *identifyCard*();

The return value is an int with the following encoding:

0 = query for member data initiated
1 = query is underway (return in one second)
2 = query complete; card owner data is in the string *memberInformation*
3 = unknown error

When the function returns < 2 the application will wait 1 second and then call again. Successful query will load the JSON representation of member data from EZ Facility into the cloud variable:

String *identifyCardResult*;

Where *identifyCardResult* is JSON formatted name:value pairs. The GUI will present each JSON name as a literal and each JSON value next to the variable name. The returned name:value pairs shall be:

- ErrorCode: 0
- ErrorMessage: OK
- Name : Bob Glicksman
- Member Number : 7654
- Card Status : Current (alternative: Revoked)
- Checkin: OK (Status of what would happen if the card was presented to a checkin station)

Note: the parameters above are mandatory. The Card Status is the result of comparing the MN card UID on the card with that in the member's record in EZ Facility. The card status is Current if they match and Revoked if they don't match.

Note that prompts for the administrator to place the card on the card writer and to subsequently remove the card will be performed by the Firmware using the hardware LCD (rather than the GUI). The LCD is right where the card is to be placed and is therefore the logical place for such user prompting.