Barcode Validation

Most scanners decode and acknowledge a barcode scan locally on the device. That is, if the scan engine can read the barcode, it will signal a good read. It is very rare that there is ever a bad read. Usually, it is either a good read, or there is NO read (barcode is too damaged or disabled).

Once the good read indication has been done for the user (beep, green light or vibration), the scanned data is sent to the host. If, for some reason, the host doesn’t like what was read, there is seemingly nothing that can be done!

Solution

To solve this problem, Socket Mobile implemented something we call Host Acknowledgement. This is a “mode” the scanner is put into via the SDK (Software Development Kit). In this mode, when the user scans a barcode, the scanner does not give any indication to the user. Instead, it sends the scanned data directly to the host. The host can then see if the scanned barcode meets any number of criteria, including sending it to a cloud service for further verification.

While the host is validating the scanned barcode, the trigger on the scanner is locked, preventing the user from scanning any other barcodes until the host acknowledges the scan to the scanner. If the Host does not acknowledge the scan, the user must wait roughly 2 minutes for the scanner to automatically unlock and become usable again (this value is adjustable and can be changed by the SDK).

Once the host software validates the barcode, it will send a command to the scanner to either do a “bad scan” notification or a “good scan” notification, and of course the notifications can include any combination of beeps (sounds), vibrations or LEDs.

Some things the host may want to validate:
- Whether the barcode is a Symbology we want (Code128, Code39, PDF417)
- Whether the barcode data is a format we are looking for
- Whether we have a valid connection to our database (the cloud)
- Whether the contents of the data meet some predetermined criteria

The SDK contains properties for setting all of this up. The properties required are:
- kSktScanPropIdDataConfirmationMode - to set scanapi mode
- kSktScanPropIdLocalAcknowledgmentDevice - to set the scanner acknowledgement mode
- kSktScanPropIdLocalDecodeActionDevice - to turn off local scan signaling
- kSktScanPropIdDataConfirmationDevice - sent after a decode event to have the scanner acknowledge a scan (good or bad). This is the only one that is sent after every decode (scan) event.

The order the first 3 properties get sent does not really matter, but the order shown above is a good reference.
There is a sample on our GitHub site (https://Github.com/socketmobile) that shows a simple way to accomplish this. This sample is setup with a 3 second delay after getting the decode event to show the trigger lock.

*We have a number of patents that cover this concept:
07686227 - Cordless Hand Scanner with Improved User Feedback
08132732 - Cordless Hand Scanner with Improved User Feedback
08235294 - Accuracy Enhanced Scanner
08702004 - Accuracy Enhanced Scanner

- Len Ott