

Version 2.0 currently has the following major goals:

- port to Unix/Linux
- 64-bit support for Microsoft Windows

In addition to these goals there are some additional items that are being proposed for the specification. One of these (Single Document Multiple Images) is currently being handled in another document. The other has to do with enhancing support around features associated with check scanning. This area isn't limited to check scanning, but it serves as the lynch pin for several items.

Based on a conversation with Harvey Spencer a check-type scanner appears to have the following flow:

#### Negotiation

- CAP\_MICRENABLED (true or false, distinct from barcodes)
- CAP\_FEEDERJOGGER (shakes input tray...arguments?)
- Multiple printer lines (do we already have this?)
- CAP\_ENDORSER (is a number enough?)
- Single Document Multiple Images (dual stream, infra-red, etc)
- CAP\_FEEDERPOCKET (output pockets, collation, strings?)
- Typical resolution: 300dpi

#### Document Types

- Checks (with MICR data)
- Credit Cards (magnetic strip)
- Licenses (magnetic strip or barcode)
- Etc...

#### Images

- Color, Grayscale, Bitonal, Infra-red (cool), etc...
- SDMI

#### Metadata

- MICR (what format – string?, control codes needed? make it a blob?)
- Barcode (Datamatrix...)
- Magnetic strip data (probably a blob, unless standardized)
- Raw magnetic waveform (definite blob)
- Snippets?

#### Possible Error Conditions

- Jam (ok)
- Double Doc (ok)
- Pod open (don't have this)

Focus error (dht)  
Document too light, too dark (dht)  
Folded corner, torn corner (dht)

The application seems to have a lot of control in this sector, so CAP\_EXTENDEDCAPS may finally start to get some use, especially for CAP\_FEEDERPOCKET and the CAP\_PRINTER\* capabilities. It looks like CAP\_AUTOSCAN needs to be FALSE and the CAP\_FEEDER\* capabilities get some use.

There was also some discussion of a transaction printer for receipts. Some devices have the ability to print arbitrary text and iconic data on a roll, along with cutting the paper and ejecting it. If we decide to go down this path it looks like a DG\_PRINTER situation, since there is need to monitor ink levels and stuff as well. We can probably take a stab at the scanning side of things...I hesitate proposing a DG\_PRINTER solution unless we have an interested party to help guide us along (preferably one who is willing to do some early implementing of the proposed additions to make sure they provide what is needed).

Towards the end there was also a discussion of downloadable menu data. This feature appears in some of the slightly higher end MFP's right now. The idea is that instead of a button press the user gets to select what application to run from a touch-screen on the device.

It's not totally clear that this is a TWAIN area for push models, especially on Windows where STI/WIA already occupy that space. However, it would make sense if the application wanted to modify that menu while the TWAIN driver was active, so it seems like a good idea to chat a bit about what that would entitle; presumably some kind of mapping between a device token (string, button number, whatever) and a localized string that the user views on the device's console.