MFP Alert Proposal

Proposed RFC 2790 Host Resources MIB changes to better support MFP alerts

(1) **PROBLEM:** there is no way to represent the overall state of components within an MFP. MFPs typically have print, copy, scan, and fax capabilities connected to/coordinated by a network controller module.
PROPOSED SOLUTION: need more device types defined within 'hrDeviceType' that represent MFP functionality such as:
   a. 'hrDeviceScanner'
   b. 'hrDeviceFax'
   c. 'hrDeviceController'
   d. 'hrDeviceAssessory'
   e. 'hrDeviceFinisher'

(2) **PROBLEM:** there are no MIB objects defined to describe basic scanner hardware capabilities of an MFP
PROPOSED SOLUTION: Need a table such as 'hrScannerTable' to expose the Scanner component capabilities such as:
   a. Color capabilitiy
   b. Document handling capability
   c. Scan resolution capability
   d. Scan speed
   e. 'hrScannerDetectedErrorState' for scanner fault classification
   f. 'hrScannerStatus' for normal scan states (e.g. scan, idle, diagnostic mode, etc.)

(3) **PROBLEM:** there are no MIB objects defined to describe basic FAX hardware capabilities of an MFP
PROPOSED SOLUTION: Need a table such as 'hrFaxTable' to expose the FAX Modem component capabilities such as:
   a. Fax modem card details such as manufacturer, speed capability, resolution, protocols supported, # of phone lines exposed, etc.
      1. I'm OK w/ using RFC 1696 – Modem MIB, however, there seems to be confusion over this RFC's "Historical" status
   b. Phone number assigned to the line used by the MFP
   c. 'hrFaxModemDetectedErrorState' for FAX modem fault classification
   d. 'hrFaxModemStatus' for normal FAX states (e.g. send, receive, line idle, disconnected, etc.)

(4) **PROBLEM:** there are no MIB objects defined to describe the hardware capabilities of any type of accessory (e.g. payment/security/accounting/etc) that can be connected to an MFP
PROPOSED SOLUTION: Need a table such as 'hrDeviceAccessoryTable' to expose the exact identity and capabilities of any external accessories which are typically connected to MFPs such as smart card readers, coin collector/card payment, accounting code entry keypad, etc. This table should comprehend attributes such as:
   a. Connection interface used (e.g. USB, parallel, serial, proprietary, etc.)
   b. Accessory device type (e.g. security, accounting, payment, diagnostic, etc.)
   c. Manufacturer of device
   d. Model Number of device
   e. Serial Number of device
   f. 'hrAccessoryDetectedErrorState'
   g. 'hrAccessoryStatus' (e.g. connected, validating, diagnostic mode, etc.)

(5) **PROBLEM:** Many times, people assume that a technician needs to be dispatched when the 'Service Requested' bit is generated by an MFP but that is simply not true is all cases.
PROPOSED SOLUTION: "serviceRequested" bit needs to be renamed to avoid confusion. A more appropriate name for this bit should probably be "interventionRequired"

(6) **PROBLEM:** All MFPs have some type of network controller which manages the various network protocols and services supported on that device. Unfortunately, the 'hrPrinterDetectedErrorState' object does not represent the various error states of the network controller.
PROPOSED SOLUTION: Need a table such as 'hrControllerTable' which contains:
   a. a new 'hrControllerDetectedErrorState' object for fault classification
   b. a new 'hrControllerStatus' object for normal controller states (e.g. idle, job processing, diagnostic mode, etc.)

(7) **PROBLEM:** The current 15 bits defined for 'hrPrinterDetectedErrorState' do not provide adequate coverage for the many machine conditions that can be generated on a MFP. Often times, "Offline" and "Service Requested" need to be assigned to most machine states because more appropriate bits to better describe the condition simply do not exist.
PROPOSED SOLUTION: If 'hrScannerDetectedErrorState', 'hrFaxModemDetectedErrorState', 'hrControllerDetectedErrorState', and 'hrAccessoryDetectedErrorState' objects are not implemented, then more bits need to be added to 'hrPrinterDetectedErrorState' to comprehend the following machine conditions
   a. external communication fault
   b. internal hardware failure
   c. internal software failure
   d. initialization / feature enablement fault
   e. job-processing-related failure
   f. threshold-related failure
   g. addressing problem
   h. resources required

Proposed RFC 3805 Printer MIB changes to better support MFPs

(1) **PROBLEM:** The 'prtAlertCodeTC' does not contain enough enumerations to cover the type of faults that occur on MFPs; 'subunitRecoverableFailure(29)', 'subunitUnrecoverableFailure(30)', 'other(1)' or 'unknown(2)' are simply not appropriate to describe several common faults on MFPs
PROPOSED SOLUTION: The following enums need to be added to the 'prtAlertCodeTC':
   a. subunitInternalCommunicationFailure(39) – faults can occur during an operation between the various MFP internal module components such as the scanner, network controller, printer, finisher, accessory, etc.
   b. subunitExternalCommunicationFailure(40) – faults can occur during an MFP outbound operation to external devices such as phone line for FAX, USB port for direct-connect scanning, SMTP / POP3 / DHCP / BOOTP / DNS / LDAP / SNTP / Netware servers, etc.
   c. subunitAuthenticationFailure(41) - access-restriction-related faults can occur between MFPs and external devices such as SMTP servers, POP3 server, DHCP servers, BOOTP servers, DNS servers, LDAP servers, etc.
   d. subunitMechanicalComponentFailure(42) – there is no generic error for handling vendor-specific MFP hardware-related faults; the current set of enums (motor failure, thermister failure, etc.) in some cases are too specific and cannot be used to describe these types of vendor-specific hardware-related faults condition
   e. subunitControlBoardFailure(43) – there is no generic MFP fault which clearly represents failures with Printed Wire Boards (PWB) or components within these PWBs; typically used for printers, scanners, and finisher modules
   f. subunitSoftwareModuleFailure(44) – faults are generated for software error conditions which run the various operations within an MFP or are processes running on the network controller
   g. subunitJobProcessingFailure(45) – faults can occur w/ various types of jobs submitted to MFPs which may disable the device (e.g. copy jobs, scan jobs, network print jobs, Fax processing jobs, etc.)
   h. subunitInitializationFailure(46) – the 'powerUp(503)' enum is too vague for some MFP vendor-specific module initialization faults which may not occur at machine power on
   i. subunitActiviationFailure(47) – faults can occur when a particular MFP feature cannot be enabled for use
   j. subunitDetectionFailure(48) – this condition is NOT the same as 'subunitMissing(9)' because a component can be physically installed in the machine but not detected by the machine; 'subunitMissing(9)' implies that the component is NOT installed in the machine

k. subunitThresholdFailure(49) – faults can occur where a normal operating range for a component cannot be achieved which is not covered by the existing 'subunitOverTemperature(36)' or 'subunitUnderTemperature(35)' enums
l. subunitFan(50) – since there are so many types of motors within the machine, a new enum is needed to clearly identify fans from those other types of motors.
m. subunitNetworkAddressFailure(51) - duplicate address-related errors occur on MFPs such as duplicate IP address, DNS name, machine SMB hostname, etc.

(2) **PROBLEM:** The 'prtConsoleDisplayBufferTextTable' does not provide any capabilities for remote applications to expose the contents of a Graphical User Interface which is present on almost all MFPs today. Most MFPs have a status region where text-based status message appear. There are also one or more sub-windows where additonal messages & graphics are exposed to the walk-up user. This data cannot be easily replicated within a remote application using the limited structure of the 'prtConsoleDisplayBufferTextTable'.
PROPOSED SOLUTION: Need to provide a way for remote applications to replicate all of the functionality currently exposed by the local UI; perhaps the following requirements are necessary
a. Need a way to expose the various UI screens for remote display purposes (e.g. provide link/address to local UI screen currently on display)
b. Need a local UI type object: GUI vs. text-based

(3) **PROBLEM:** There is no way to represent nor control the various buttons (NOT the keyboard if present) which exist on MFP consoles today.
PROPOSED SOLUTION: Need to expose a 'prtConsoleButtonTable' to define and control all of the buttons on the UI

(4) **PROBLEM:** Because of the large numer of fault conditions that can occur within an MFP, many vendors use some type of code (e.g. status code, fault code, etc.) to uniquely identify each fault condition. Unfortunately, this fault code is not exposed within an industry standard MIB object. As a result, vendors typically use different 'prtAlertTable' objects to expose this data which lead to non-coherent implementations.
PROPOSED SOLUTION: An object needs to be added to the 'prtAlertTable' to uniquely identify each fault codition; perhaps called 'prtAlertFaultCode' which is a string of 10-20 alphanumeric characters including dashes and dots, etc.

(5) **PROBLEM:** Each fault exposed within the 'prtAlertTable' may have a different repair action required to resolved the condition. Unfortunately, there is no MIB object defined today to expose this type of data.
PROPOSED SOLUTION: An object needs to be added to the 'prtAlertTable' to define any actions that need to be performed in order to resolve the fault codition; perhaps called 'prtAlertRepairAction' which is a string of 255 characters (e.g. "remove the toner cartridge, shake it a couple of times, then replace it back into the machine", "reboot the machine", "call for service, if the fault persists", etc.)

(6) **PROBLEM:** A fault may impact one or more services within a MFP. Unfortunately, there is NO service table that exists within an industry standard MIB which defines the impact of a fault on the machines core functions such as print, copy, network scan, embedded fax, etc.
PROPOSED SOLUTION: An object needs to be added to the 'prtAlertTable' to expose the impact of the fault on the rest of the MFP based services (e.g. copy, print, fax, scan, etc.); perhaps call 'prtAlertServiceImpact' which could be a bitwise string where each bit that is set represents a disabled service

(7) **PROBLEM:** RFC 3805 does NOT provide enough implementation details regarding the intended implementation of the 'prtMarkerLifeCount' object. I have seen various implementations of this MIB object across several vendors.
PROPOSED SOLUTION: Need to exposed more explicit details regarding the implementation of the 'prtMarkerLifeCount' object per Ira McDonald's 7/24/05 response:

```
-----Original Message-----
From: McDonald, Ira [mailto:imcdonald@sharplabs.com]
Sent: Sunday, July 24, 2005 12:23 PM
To: Silver, Thomas; McDonald, Ira; harryl@us.ibm.com;
ron.bergman@hitachi-ps.us.com; Zehler, Peter; 'pmp@pwg.org'
Subject: RE: Need clarification on the definition of RFC 3805
'prtMarkerLifeCount' object
```

```
Hi Tom,

Sorry I missed this the first time around.  Wasn't sent to PMP mailing
list, so it got killed by spam filters.

The answer to your question is that both behaviors by duplex printers on
a single page job are historically correct (increment by one or
increment by two).  But your question only makes sense if the
PrtMarkerCounterUnitTC chosen unit is 'impressions(7)'.

The principal use of PrtMarkerLifeCount is to record use of the marker
physical path.  A duplex but blank back side _probably_ still went
through the duplex path and caused wear on rollers, etc.

There is new guidance here.  In the PWG Imaging System Counters spec
(completed last call and soon to be formally approved), a 'blank
impression' MUST be counted in an overall 'Impressions'
counter (and also in the separate 'BlankImpressions' counter).
Therefore, the best practice for prtMarkerLifeCount using impressions
would now be to increment by TWO (not intuitive, I know).

Pete Zehler - please put in your two cents here, since it's a question
from Xerox - thanks.

Cheers,
- Ira

Ira McDonald (Musician / Software Architect) Blue Roof Music / High
North Inc PO Box 221  Grand Marais, MI  49839
phone: +1-906-494-2434
email: imcdonald@sharplabs.com
```

(8) **PROBLEM:** RFCs 3805 & 3806 do NOT provide MIB objects for exposing the part number for a consumable that needs to be replaced within an MFP. When a consumable depletion condition occurs, the machine administrator or service provider has to determine the name of the consumable which is depleted and then look-up a corresponding part number for that consumable because the part number is typically not exposed via the MIBs on most MFPs. This part number look-up task may be time-consuming for those that don't have good supplies management processes in place.
PROPOSED SOLUTION: An object needs to be added to the 'prtMarkerSuppliesTable' to expose the consumable part number; perhaps call 'prtMarkerSuppliesPartNumber' which is a string of 255 characters.
PROPOSED SOLUTION: An object needs to be added to the 'finSupplyTable' to expose the consumable part number; perhaps call 'finSupplyPartNumber' which is a string of 255 characters.

(9) **PROBLEM:** RFCs 3805 & 3806 do NOT provide MIB objects for exposing the install date of a consumable within an MFP. Machine administrators may buy extra consumables to ensure that when a machine runs out of a consumable, a replacement is available and on-hand to restore the machine to normal operation. However, having extra consumables lying around is prone to theft and wastes money in terms of the consumable items, the real-estate costs to store the consumables, and the headache associated w/ managing/securing the consumable. The consumable install date can be used to implement a "just-in-time" consumable reordering process.
PROPOSED SOLUTION: An object needs to be added to the 'prtMarkerSuppliesTable' to expose the consumable install date; perhaps call 'prtMarkerSuppliesInstallDate' which is a string of syntax 'DateAndTime'
PROPOSED SOLUTION: An object needs to be added to the 'finSupplyTable' to expose the consumable install date; perhaps call 'finSupplyInstallDate' which is a string of syntax 'DateAndTime'.