MPOST Integration Manual

API Specification and Best Practices

MEI

Table of Contents

1 Change History

Issue	Date	Description
1.0	08/31/2011	Initial Revision
	10/26/2011	Added comment about Download Restart as an exceptional condition.
2.0	01/24/2012	Added support for CashBox Cleanliness Reporting.
3.0	07/30/2012	Added support for “Clear Audit” command. SC Advance only.
4.0	08/02/2012	Added support for retrieving Audit structure of all banknotes.
5.0	02/13/2013	Added support for new property “LastBNFError”
	02/20/2013	Added warning to calibration command that proper paper is required.
6.0	01/13/2014	Updated references for V2.90 Release.
7.0	11/26/2014	Added support for GetRFIDSerialNumber in V3.40.
8.0	06/08/2015	Added a new STACKED event with document type and value (STACKED_WITH_DOC_INFO)
9.0	07/01/2015	Added Protocol Level enumeration, IDocument interface and Barcode class; created a new chapter for Document related interface and classes; updated obsolete calibration feature description.

2 Introduction

2.1 Acronyms and Definitions

Keyword	Meaning / Context
API	Application Programming Interface. Refers to the MPOST API
M/POST or MPOST	MEI Point Of Service Toolkit. This refers to the Library used to communicate with MEI Bill acceptor products.
EBDS	Enhanced Bi-Directional Serial – This is the MEI Specific protocol used to communicate with some MEI bill acceptors. Most of this complexity is abstracted away from the user through MPOST.
Device	Refers to the bill acceptor
Document	General description for one of the following types of items accepted by a bill acceptor (Banknote/Bill, Barcode, Coupon)
BNF	Bunch Note Feeder. This is an additional device that can be attached to the Cashflow SC and SCN products to allow automatic feeding of several notes at once.
Note	Refers to a Banknote. Also known as a Bill.
Host	Refers to the hardware and software using the MPOST API.

2.2 Scope

The scope of this document is to provide a detailed description of the MPOST Library and how to use it. This document will detail all of the available properties and methods of the API in addition to some recommended use cases.

A more detailed description of many of the concepts and settings described in this document can be found in the EBDS Protocol Specification Document. Please contact our technical support to acquire the latest version of this document if desired.

2.3 Requirements

2.3.1 Products

The MPOST library is designed to work with any EBDS supported bill acceptor from MEI. This includes the following products: Series 2000 bill acceptor, Cashflow SC and SC advance devices.

2.3.2 Software

MPOST comes in a few different versions to support a wider number of host integrations on multiple operating systems. Please see section 3.1 Overview to see which version of MPOST would best suite your needs.

2.3.3 Hardware

The computer that will be using MPOST must also have either a serial port or a USB port. USB support is only available on Windows Operating system through a USB to Serial Communications Driver.

3 MEI Point of Service Toolkit (M/POST)

The M/POST API provides for significant savings of time and effort for coding, debugging and deploying applications that require the acceptance and handling of cash. The M/POST is based on the Properties/Methods/Events model of programming used in objected oriented systems. The following section will outline the specific classes, properties and methods of the MPOST API.

3.1 Overview

M/POST is currently released in two stable libraries that support a wide range of applications. The following table displays the current and previous libraries.

Name	Description	Status
M/POST .NET	Library written in C# and supports the .NET framework 2.0 or greater. The DLL can also be exposed to COM projects to support older languages like VB6.	Stable, active project. Current version as of June 2015 is V3.50
M/POST Java	Full java implementation that supports Windows and Linux. The release package contains all the files necessary for serial communications on both operating systems.	Stable, active project. Current version as of July 2015 is V3.50
M/POST OLE	Library written in VB6.	Deprecated. No new development planned for this version. Last released version was V1.23 during Nov 2010. Recommended upgrade - .NET with COM operability
M/POST for Linux	Aimed at Linux platforms and supported the C and C++ languages	Obsolete. No development since V1.00 May 2008. Recommended upgrade – Java.

3.1.1 M/POST Library differences

Since there are libraries written in different programming languages, there are slight differences in the properties and methods of the objects. Every effort was made to make the libraries as syntactically similar as possible but some general differences are listed below and other differences will be listed as described:

Item	.NET	Java
Object Properties	Full support of properties. (ex. acceptor.Connected)	No support for properties. All properties are methods prefixed with “get” or “set” depending on the operation. (ex. acceptor.getConnected())
Enumerations	Enumerations are contained inside the Acceptor Class.	Enumerations exist in their own files located in the com.meiglobal.ebds.api.pub package

3.2 M/POST Enumerations

The M/POST libraries make use of enumerations for return values on occasions. These enumerations are listed below

3.2.1 BNFStatus Enumeration

This type describes possible statuses for the BNF device.

Type	Description
Unknown	Unknown status. Reported by devices that do not support the BNF status command.
OK	Attached BNF is operating normally.
NotAttached	No BNF is attached.
Error	A condition has resulted in the BNF not working properly. Please review the LastBNFError property to determine the root cause.

3.2.2 BNFErrorStatus Enumeration

The BNF can be placed out of service due to several exceptional cases. The BNF Error status will reflect the root cause of the out of service exception.

Type	Description
NoError	BNF is operating normally.
MotorStall	Usually a mechanical failure, a motor stall means the BNF was unable to transport a document for an unknown reason. This can also occur if a very thick obstruction is inserted causing the motors to lock up.
CartridgeRemoved	The BNF cartridge has been removed or is not seated correctly. Please reset the cartridge.
StubOut	The BNF was unable to draw in the supplied documents after five attempts. The documents should be removed and re-fed into the BNF.
CoveredStubOut	The BNF was unable to draw in a document properly after five attempts and an obstruction is detected in the BNF bill path. This is a more severe version of the Stub Out condition.
TooManyRejects	The BNF can be configured to stop operations if too many rejections are detected. To correct this condition, remove and reseat the current documents in the feed tray.

3.2.3 CashBoxCleanlinessEnum Enumeration

This type describes possible cashbox maintenance events. This only applies to hosts that have enabled the reporting functionality through the function call EnableCashboxCleanlinessReported().

Type	Description
CleaningRecommended	A cashbox has been inserted that is approaching critical sensor levels. The device will continue to operate normally but the cashbox should be addressed before any future use.
CleaningRequired	A cashbox has been inserted that is currently unusable due to poor sensor levels. The device will NOT enter normal operations and the cashbox must be addressed before it can be used.

3.2.4 DocumentType Enumeration

This type describes the type of document currently being processed.

Type	Description
None	No document is currently being processed.
NoValue	A document with no value is being processed.
Bill	A bank note is being processed.
Barcode	A barcode is being processed.
Coupon	A generic coupon is being processed.

3.2.5 Orientation Enumeration

This type describes the orientation of a note.

Orientation
RightUp
RightDown
LeftUp
LeftDown
Unknown (.Net) / UknownOrientation (Java)

3.2.6 OrientationCtl Enumeration

This type describes the ways that the orientation of notes may be controlled.

	Accepted Note Orientation			
OrientationControl	Right-Up	Right-Down	Left-Up	Left-Down
FourWay	✓	✓	✓	✓
TwoWay	✓		✓	
OneWay	✓			

3.2.7 PowerUp Enumeration

The Power Up enumeration is used primarily at startup to tell the device in which mode to operate.

PowerUp	Notes
A	Default. Recommended
B	
C	
E	(Deprecated – Not used)

3.2.8 Protocol Level Enumeration

Protocol Level	Notes
Standard	Standard communication speed at 9600bps
Fast_Download_Serial	Fast communication speed at 38400bps for download

3.2.9 State Enumeration

This enumeration is meant to be used by the Acceptor to return the current device state.
State = { Disconnected, Connecting, PupEscrow, Idling, Accepting, Escrow, Stacking, Stacked, Returning, Returned, Rejected, Jammed, Stalled, Failed, CalibrateStart, Calibrating, DownloadStart, DownloadRestart, Downloading }

State	Description
Disconnected	Not connected to a device
Connecting	Device is connected (Deprecated)
PupEscrow	There is a document at escrow after a power up
Idling	Device is idle
Accepting	Device is accepting a note
Escrow	There is a document at escrow
Stacking	Device is staking a document.
Stacked	Device has successfully stacked a document. (It can no longer be returned/ retrieved by the user)
Returning	A document is being returned to the user
Returned	A document has been successfully returned to the user.
Rejected	A document was not valid and automatically rejected by the device
Jammed	The device is jammed and will not continue normal operations until this situation is resolved
Stalled	The device is stalled and will not continue normal operations until this situation is resolved
Failed	The device has entered a failure mode and will not continue normal operations until this situation is resolved. This usually requires a device restart.
CalibrateStart	Device is being configured to enter the calibration mode
Calibrating	The device is currently being calibrated.
DownloadStart	Device is being configured to enter the download mode. Used to flash new software to the device.
DownloadRestart	Device has requested to enter download mode (This occurs if there was an error during a previous download). The device will not operate normally until it has completed the download successfully.
Downloading	The device is currently downloading new software

3.2.10 Bezel Enumeration

Used when setting the Bezel type.

Bezel
Standard
Platform
Diagnostic

3.3 M/POST Acceptor Object Properties

The following section outlines the properties available for the main object of the M/POST Library. Please reference M/POST Document Interface and Classes and M/POST Auxiliary Object Properties and Methods for the properties of the secondary objects.

3.3.1 Acceptor Properties - Overview

Name	Type	Version	Access	When Usable
ApplicationID	STRING	1.00	R	CapApplicationID
ApplicationPN	STRING	1.00	R	CapApplicationPN
AssetNumber	STRING	2.00	R/W	CapEasitrax
AuditLifeTimeTotals	INT32[ ]	1.00	R	CapAudit.
AuditPerformance	INT32[ ]	1.00	R	CapAudit.
AuditQP	INT32[ ]	1.00	R	CapAudit.
AutoStack	BOOLEAN	1.00	R/W	Read = Always Write = Connected.
BarCode	STRING	1.00	R	DocumentType == Barcode.
Bill	Bill	1.00	R	DocumentType == Bill.
BillTypeEnables	BOOLEAN[ ]	1.00	R/W	Connected.
BillTypes	Bill[ ]	1.00	R	Connected.
BillValueEnables	BOOLEAN[ ]	1.00	R/W	Connected.
BillValues	Bill[ ]	1.00	R	Connected.
BNFStatus	BNFStatus	1.00	R	CapBNFStatus
BootPN	STRING	1.00	R	CapBootPN.
CapAdvBookmark	BOOLEAN	2.00	R	Connected.
CapApplicationID	BOOLEAN	1.00	R	Connected.
CapApplicationPN	BOOLEAN	1.00	R	Connected.
CapAssetNumber	BOOLEAN	1.00	R	Connected.
CapAudit	BOOLEAN	1.00	R	Connected.
CapBarCodes	BOOLEAN	1.00	R	Connected.
CapBarCodesExt	BOOLEAN	1.00	R	Connected.
CapBNFStatus	BOOLEAN	1.00	R	Connected.
CapBookmark	BOOLEAN	1.00	R	Connected.
CapBootPN	BOOLEAN	1.00	R	Connected.
CapCalibrate	BOOLEAN	1.00	R	Connected.
CapCashBoxTotal	BOOLEAN	1.00	R	Connected.
CapCearAudit	BOOLEAN	2.60	R	Connected.
CapCouponExt	BOOLEAN	1.00	R	Connected.
CapDevicePaused	BOOLEAN	1.00	R	Connected.
CapDeviceSoftReset	BOOLEAN	1.00	R	Connected.
CapDeviceType	BOOLEAN	1.00	R	Connected.
CapDeviceResets	BOOLEAN	1.00	R	Connected.
CapDeviceSerialNumber	BOOLEAN	1.00	R	Connected.
CapEasitrax	BOOLEAN	2.00	R	Connected.
CapEscrowTimeout	BOOLEAN	1.00	R	Connected.
CapFlashDownload	BOOLEAN	1.00	R	Connected.

<Table Continues on Next Page>

< Continuation from Previous Page - Acceptor Properties Overview>

Name	Type	Version	Access	When Usable
CapNoPush	BOOLEAN	1.00	R	Connected.
CapNoteRetrieved	BOOLEAN	2.00	R	Connected.
CapOrientationExt	BOOLEAN	1.00	R	Connected.
CapPupExt	BOOLEAN	1.00	R	Connected.
CapSetBezel	BOOLEAN	1.10	R	Connected.
CapTestDoc	BOOLEAN	1.00	R	Connected.
CapVariantID	BOOLEAN	1.00	R	Connected.
CapVariantPN	BOOLEAN	1.00	R	Connected.
CashBoxAttached	BOOLEAN	1.00	R	Connected.
CashBoxFull	BOOLEAN	1.00	R	Connected.
CashBoxTotal	INT32	1.00	R	CapCashBoxTotal.
Connected	BOOLEAN	1.00	R	Always.
Coupon	Coupon	1.00	R	DocumentType == Coupon
DebugLog	BOOLEAN	1.00	R/W	Always.
DebugLogPath	STRING	1.00	R/W	Read: Always Write: When DebugLog is false.
DeviceBusy	BOOLEAN	1.00	R	Connected.
DeviceCRC	INT32	1.00	R	Connected.
DeviceFailure	BOOLEAN	1.00	R	Connected.
DeviceJammed	BOOLEAN	1.00	R	Connected.
DeviceModel	INT32	1.00	R	Connected.
DevicePaused	BOOLEAN	1.00	R	CapDevicePaused.
DevicePortName (getPort in Java)	STRING	1.00	R	Connected.
DevicePowerUp	PowerUp	1.00	R	Connected.
DeviceResets	INT32	1.00	R	CapDeviceResets.
DeviceRevision	INT32	1.00	R	Connected.
DeviceSerialNumber	STRING	1.00	R	CapDeviceSerialNumber.
“	“	1.15	R	Connected
DeviceSignature	INT32	1.00	R	Connected.
DeviceStalled (.NET only)	BOOLEAN	1.00	R	CapNoPush.
DeviceState	State	1.00	R	Always.
DeviceType	STRING	1.00	R	CapDeviceType
DisconnectTimeout	INT32	2.00	R/W	Always.
DocType	DocType	1.00	R	After DeviceState == Escrow Or DeviceState == Stacked
EnableAcceptance	BOOLEAN	1.00	R/W	Connected.
EnableBarCodes	BOOLEAN	1.00	R/W	CapBarCodes.
EnableBookmarks	BOOLEAN	1.00	R/W	CapBookmark.
EnableCouponExt	BOOLEAN	1.00	R/W	CapCouponExt
EnableNoPush	BOOLEAN	1.00	R/W	CapNoPush.

<Table Continues on Next Page>

<Continuation from Previous Page - Acceptor Properties Overview>

Name	Type	Version	Access	When Usable
EscrowOrientation	Orientation	1.00	R	CapOrientationExt && After DeviceState == Escrow.
HighSecurity	BOOLEAN	1.00	R/W	Connected.
LastBNFError	BNFErrorStatus	2.80	R	CapBNFStatus
OrientationCtl	OrientationCtl	1.00	R/W	Connected.
OrientationCtlExt	OrientationCtl	1.00	R/W	CapOrientationExt.
VariantID	STRING	1.00	R	CapVariantID.
VariantNames	STRING[ ]	1.00	R	Connected.
VariantPN	STRING	1.00	R	CapVariantPN.
APIVersion (.NET) getVersion (Java)	STRING	1.00	R	Always

3.3.2 Acceptor Properties – Details

This section provides details on each of the Acceptor Object's properties.

Name	Return	Version	Access	When Usable
ApplicationID	STRING	1.00	R	CapApplicationID
Read Point:	Queries Acceptor	**Write Point :**		n/a

Description: This property contains the application ID. Will throw an exception if not connected, CapApplicationID is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
AssetNumber	STRING	2.00	R/W	CapEasitrax
Read Point:	Queries Acceptor	**Write Point :**		Sent to the device on the next outbound message.

Description: This property contains the device's asset number. This is a configurable setting that can be used by the host organization in order to uniquely identify this device. Will throw an exception when reading if not connected, CapEasitrax is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }.* Will throw an exception when writing if not connected, CapEasitrax is false, parameter is greater than 16 characters or the DeviceState is not either Idle or Failed.

Name	Return	Version	Access	When Usable
ApplicationPN	STRING	1.00	R	CapApplicationPN
Read Point:	Queries Acceptor	**Write Point :**		n/a

Description: This property contains the application part number. Will throw an exception if not connected, CapApplicationPN is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
AuditLifeTimeTotals	INT32[]	1.00	R	CapAudit
Read Point:	Queries Acceptor	**Write Point :**		n/a
Description:	This property contains an array of integers with the lifetime data. Will throw an exception if not connected, CapAudit is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*			

Name	Return	Version	Access	When Usable
AuditPerformance	INT32[]	1.00	R	CapAudit
Read Point:	Queries Acceptor	**Write Point :**		n/a
Description:	This property contains an array of integers with the performance data. Will throw an exception if not connected, CapAudit is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*			

Name	Return	Version	Access	When Usable
AuditQP	INT32[]	1.00	R	CapAudit
Read Point:	Queries Acceptor	**Write Point :**		n/a
Description:	This property contains an array of integers with the performance data. Will throw an exception if not connected, CapAudit is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*			

Name	Return	Version	Access	When Usable
AutoStack	BOOLEAN	1.00	R/W	Connected
Read Point:	Local API variable	**Write Point :**		Immediate
Description:	This property controls the handling of documents. If set, documents are automatically stacked when they are received. If not set, the application is informed via the ESCROW event when a document arrives. Will throw an exception if not connected.			

Name	Return	Version	Access	When Usable
Barcode	STRING	1.00	R	DocumentType == Barcode
Read Point:	At barcode escrow	**Write Point :**		n/a
Description:	This property contains the barcode info extracted from the most recent bar-coded document. Will throw an exception if the DocumentType is not of type Barcode.			

Name	Return	Version	Access	When Usable
Bill	Bill	1.00	R	DocumentType == Bill
Read Point:	At bill escrow	**Write Point :**		n/a
Description:	This property contains the bill object with information extracted from the most recent bank note. Will throw an exception if the DocumentType is not of type Bill.			

Name	Return	Version	Access	When Usable
BillTypeEnables	BOOLEAN[]	1.00	R/W	Connected
Read Point:	Created on Connect. Prefilled with "TRUE"	**Write Point :**		Sent to the device on the next outbound message.

Description: Only applicable for devices that support extended note processing. This property is an array of Boolean values that correspond to the entries in the BillTypes property. A true value enables acceptance while a false value will reject those notes.

Note: Changes in the BillValueEnables property will result in changes to the BillTypesEnables property. However, changes made to the BillTypesEnables do NOT propagate back to the BillValueEnables property. In general, the application should use one of the bill enable properties and not switch back and forth between them as this may cause unexpected results.

Name	Return	Version	Access	When Usable
BillTypes	Bill[]	1.00	R	Connected
Read Point:	Cached on connect	**Write Point :**		n/a

Description: This property is an array of all acceptable notes for the device. This includes entries for each variety of banknotes. This table is constructed when a connection to the acceptor is established.

An example of a truncated print out of this property for a bill acceptor loaded with the US dollar variant part number 490320223 is shown below:

Note Index	Country	Value	Version Identfiers
1	USD	1	C A B B
2	USD	2	C A B A
3	USD	2	C B B A
4	USD	5	C A B B
5	USD	5	D A B C
6	USD	10	D A B B
…	…	…	…

Name	Return	Version	Access	When Usable
BillValueEnables	BOOLEAN[]	1.00	R/W	Connected
Read Point:	Created on Connect. Prefilled with "TRUE"	**Write Point :**		Sent to the device on the next outbound message.

Description: Only applicable for devices that support extended note processing. This property is an array of Boolean values that correspond to the entries in the BillValues property. Acceptance of bills that correspond to those values may be controlled with true entries accepting notes and false entries rejecting them.

Note: Changes in the BillValueEnables property will result in changes to the BillTypesEnables property. However, changes made to the BillTypesEnables do NOT propagate back to the BillValueEnables property. In general, the application should use one of the bill enable properties and not switch back and forth between them as this may cause unexpected results.

Name	Return	Version	Access	When Usable
BillValues	Bill[]	1.00	R	Connected
Read Point:	Cached on connect	**Write Point :**		n/a

Description: This property is an array of all acceptable notes for the device. Variations in like-valued banknotes are ignored so that each entry is a unique value or country. This table is constructed when a connection to the acceptor is established.

An example of a print out of this property for a bill acceptor loaded with the US dollar variant part number 490320223 is shown below:

Note Index	Country	Value	Version Identifiers (not used in this property)
1	USD	1	* * * *
2	USD	2	* * * *
3	USD	5	* * * *
4	USD	10	* * * *
5	USD	20	* * * *
6	USD	50	* * * *
7	USD	100	* * * *

Note that the view of the bill set is simpler than that in the BillTypes property, but that the detail information about the various bill types and variations is not available.

Name	Return	Version	Access	When Usable
BNFStatus	BNFStatus	1.00	R	CapBNFStatus
Read Point:	Queries Acceptor	**Write Point :**		n/a

Description: This property can be used to determine the status of the optional bunch note feeder attachment. This property is only valid when CapBNFStatus is true. Will throw an exception if not connected, CapBNFStatus is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
BootPN	STRING	1.00	R	CapBootPN
Read Point:	Queries Acceptor	**Write Point :**		n/a

Description: This property string contains the boot part number. Will throw an exception if not connected, CapBootPN is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
Cap***	BOOLEAN	Varying	R	Connected
Read Point:	Cached on connect	**Write Point :**		n/a

Description: Properties beginning with "Cap" are all capability flags that indicate what features are present on the connected device. For the most part, these flags are derived from the model number field sent by the acceptor

Name	Model #	Other Criteria
CapAdvBookmark		Device Capability Map is set for Advanced Bookmark Functionality.
CapApplicationID		Device Capability Map is set for query acceptor application ID.
CapApplicationPN	"T", "U"	
CapAssetNumber	"T", "U"	
CapAudit	"T", "U"	
CapBarCodes	15, 23 , "T", "U"	
CapBarCodesExt		The RawBarcodes bit is set in device replies
CapBNFStatus		Device Capability Map is set for query BNF status.
CapBookmark		True (Always)
CapBootPN	"T", "U"	
CapCalibrate		True (Always)
CapCashBoxTotal	"A", "B", "C", "D", "G", "M", "P", "W", "X"	
CapClearAudit		SC Advance models only.
CapCouponExt	"P", "X"	

<Table Continues on Next Page>

<Continuation from Previous Page>

Name	Model #	Other Criteria
CapDevicePaused	31, “P”, “X”	
CapDeviceSoftReset	31, “A”, “B”, “C”, “D”, “G”, “M”, “P”, “T”, “U”, “W”, “X”	
CapDeviceType	“T”, “U”	
CapDeviceResets	“A”, “B”, “C”, “D”, “G”, “M”, “P”, “T”, “U”, “W”, “X”	
CapDeviceSerialNumber	“T”, “U”	Deprecated. Capability flag no longer required as of V1.15. The device will return empty string if DeviceSerialNumber is not supported
CapEasitrax		Device Capability Map is set for query Easitrax.
CapEscrowTimeout	“T”, “U”	
CapFlashDownload		True (Always)
CapNoPush	23, 31, “P”, “X”	
CapNoteRetrieved		Device Capability Map is set for Note Retrieved Functionality.
CapOrientationExt		Device Capability Map is set for extended orientation control.
CapPupExt		Device Capability Map is set for extended PUP mode. (Deprecated – always false)
CapSetBezel		Device Capability Map is set for set bezel.
CapTestDoc		Device Capability Map is set for test document accepted.
CapVariantID		Device Capability Map is set for query acceptor variant ID.
CapVariantPN	“T”, “U”	

Name	Return	Version	Access	When Usable
CashboxAttached	BOOLEAN	1.00	R	Connected
Read Point:	Updated on poll	**Write Point :**		n/a

Description: This property is true when a cash box is attached to the bill acceptor.

Name	Return	Version	Access	When Usable
CashboxFull	BOOLEAN	1.00	R	Connected
Read Point:	Updated on poll	**Write Point :**		n/a

Description: This property is true when a cash box is full and can no longer accept bills.

Name	Return	Version	Access	When Usable
CashBoxTotal	INT32	1.00	R	Device not busy. CapCashBoxTotal
Read Point:	Queries Acceptor	**Write Point :**		n/a

Description: This property returns the amount of currency believed to be present in the cashbox. Will throw an exception if not connected, CapCashBoxTotal is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Note that if this property is referenced after the cashbox is removed (for example, in either the CASHBOX_REMOVED or CASHBOX_INSTALLED event handlers) then the cash box value in the bill acceptor will be cleared so that the next time this property is referenced, it will have a value of zero.

Name	Return	Version	Access	When Usable
Connected	BOOLEAN	1.00	R	Always
Read Point:	API status variable	**Write Point :**		n/a

Description: The connected property is true if a device is connected and responding to the host system. This becomes false when the device does not respond to polls for more than an allowable limit, or the connection is closed by the host. This timeout value can be adjusted using the DisconnectTimeout property.

Name	Return	Version	Access	When Usable
Coupon	Coupon	1.00	R	DocumentType == Coupon
Read Point:	At Coupon escrow	**Write Point :**		n/a

Description: This property contains the coupon info extracted from the most recent generic coupon. Will throw an exception if the DocumentType is not of type Coupon.

Name	Return	Version	Access	When Usable
DebugLog	BOOLEAN	1.00	R/W	Always
Read Point:	Local API variable	**Write Point :**		Immediate

Description: This property is used to control the generation of a debug log file. In the development phase of an application, this log can be useful in diagnosing any problems or issues that might arise. Note that the debug log file is locked while the application is using it.

Name	Return	Version	Access	When Usable
DebugLogPath	STRING	1.00	R/W	Read: Always Write: When DebugLog is false
Read Point:	Local API variable	**Write Point :**		Immediate

Description: This property is used to control the location of the debug log file. When the DebugLog property is set to true, the value of this property is used to determine the location of the log file. By default, the log file is created in the same folder where the application is deployed.

Name	Return	Version	Access	When Usable
DeviceBusy	BOOLEAN	1.00	R	Connected
Read Point:	Updated on poll	**Write Point :**		n/a

Description: Returns false if the device is in the IDLE State. This is the preferred in which to issue complex commands. If True, it is not guaranteed that the device will be able to process the command.

Name	Return	Version	Access	When Usable
DeviceCRC	INT32	1.00	R	Connected
Read Point:	Queries Acceptor	**Write Point :**		n/a

Description: This value is the CRC of the flash memory (excluding boot and "special" areas). Will throw an exception if not connected or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
DeviceFailure	BOOLEAN	1.00	R	Connected
Read Point:	Updated on poll	**Write Point :**		n/a

Description: When this flag is set, the bill acceptor is out of service and requires attention, typically in the form of a call by a service technician.

Name	Return	Version	Access	When Usable
DeviceJammed	BOOLEAN	1.00	R	Connected
Read Point:	Updated on poll	**Write Point :**		n/a

Description: When this flag is set, the bill acceptor has encountered a jam condition and requires attention. Typically there is an obstruction in the bill path.

Name	Return	Version	Access	When Usable
DeviceModel	INT32	1.00	R	Connected
Read Point:	Cached on connect	**Write Point :**		n/a

Description: This number is set from the model number field returned by the bill acceptor for each poll. If the value is between 32 and 126, it should be treated as a single printable character.

Name	Return	Version	Access	When Usable
DevicePaused	BOOLEAN	1.00	R	CapDevicePaused
Read Point:	Updated on poll	**Write Point :**		n/a

Description: This flag is set when the bill acceptor is in the PAUSED condition. This occurs when the consumer attempts to insert another bill while the previous one is still being processed. The system pauses to avoid grabbing two bills and "stealing" the second one. If the property is not supported, it will always return false.

Name	Return	Version	Access	When Usable
DevicePortName	STRING	1.00	R	Connected
Read Point:	Cached on connect	**Write Point :**		n/a

Description: This is a copy of the port_name parameter passed into the Open method. It is the serial or virtual serial port used to communicate with the bill acceptor.

Name	Return	Version	Access	When Usable
DevicePowerUp	PowerUp	1.00	R	Connected
Read Point:	Cached on connect	**Write Point :**		n/a

Description: This is a copy of the power-up parameter passed into the Open method. It is of type PowerUp and control the behavior of the bill acceptor when a bill is being processed during a power failure.

Name	Return	Version	Access	When Usable
DeviceResets	INT32	1.00	R	CapDeviceResets
Read Point:	Cached on connect	**Write Point :**		n/a

Description: This property returns the number of times that the bill acceptor has been reset. Will throw an exception if not connected, CapDeviceResets is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
DeviceRevision	INT32	1.00	R	Connected
Read Point:	Updated on poll	**Write Point :**		n/a

Description: This property is the numeric value contained in the standard EBDS revision field.

Name	Return	Version	Access	When Usable
DeviceSerialNumber	STRING	1.15	R	Connected (V1.15 or greater) *CapDeviceSerialNumber (V1.00)*
Read Point:	Queries acceptor	**Write Point :**		n/a

Description: This property is the serial number of the attached device in string format. For devices that do not support this property, the empty string is returned.

Name	Return	Version	Access	When Usable
DeviceSignature	INT32	1.00	R	Connected
Read Point:	Queries acceptor	**Write Point :**		n/a

Description: This property returns a value that corresponds to the legacy mode device signature. This signature can be used to ensure that the code contained within a unit matches the expected code.

Name	Return	Version	Access	When Usable
DeviceStalled	BOOLEAN	1.00	R	CapNoPush
Read Point:	Updated on poll	**Write Point :**		n/a

Description: This flag is set when the bill acceptor is in the STALLED condition. This occurs when a jam is detected and NoPush mode is active. The system pauses so that the operator can examine the bill that caused the jam and to check for a possible fraud.

NOTE: This property is deprecated. You should use the device state property to determine if the unit is in the Stalled state.

Name	Return	Version	Access	When Usable
DeviceState	State	1.00	R	Always
Read Point:	Updated on poll	**Write Point :**		n/a

Description: This variable reflects the current state of the bill acceptor.

Name	Return	Version	Access	When Usable
DeviceType	STRING	1.00	R	CapDeviceType
Read Point:	Queries acceptor	**Write Point :**		n/a

Description: This property is the device type of the attached device in string format. Will throw an exception if not connected, CapDeviceType is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
DisconnectTimeout	INT32	2.00	R/W	Always
Read Point:	API status variable	**Write Point :**		Immediate

Description: This timeout value determines how quickly the API will raise the Disconnected event after losing communications. The default for this value is 30 seconds. It is not recommended to set this value below 4 seconds to ensure that the device does not report disconnected incorrectly if the communication line is temporarily interrupted for a short period of time.

Name	Return	Version	Access	When Usable
DocType	DocumentType	1.00	R	After DeviceState == Escrow
Read Point:	Queries acceptor	**Write Point :**		n/a

Description: This property is the type of document currently being processed. Typically this property is interrogated at escrow to determine how it should be processed.

Name	Return	Version	Access	When Usable
EnableAcceptance	BOOLEAN	1.00	R/W	Connected
Read Point:	API status variable	**Write Point :**		Updated on poll

Description: This property is used to control acceptance of documents by the bill acceptor. If false, the bill acceptor will accept no documents, if true, then the selected bills, and other documents will be accepted. Will throw an exception if not connected.

Name	Return	Version	Access	When Usable
EnableBarCodes	BOOLEAN	1.00	R/W	CapBarCodes
Read Point:	API status variable	**Write Point :**		Updated on poll

Description: This property is used to control acceptance of bar coded documents. If false, the bill acceptor will accept no bar codes, if true, then bar codes will be accepted if the EnableAcceptance property is true. Will throw an exception if not connected, CapBarCodes is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
EnableBookmarks	BOOLEAN	1.00	R/W	CapBookmark
Read Point:	API status variable	**Write Point :**		Updated on poll

Description: This property is used to control acceptance of book mark documents. If false, the bill acceptor will accept no book marks. If true, book marks will be accepted if the EnableAcceptance property is true. Will throw an exception if not connected, CapBookmark is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
EnableCouponExt	BOOLEAN	1.00	R/W	CapCouponExt
Read Point:	API status variable	**Write Point :**		Updated on poll

Description: This property is used to control acceptance of generic coupon documents. If false, the bill acceptor will treat the coupons the same as a bill of the same value. If true, coupons receive special processing and more details about the coupon are available by way of the Coupon property. Will throw an exception if not connected, CapCouponExt is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
EnableNoPush	BOOLEAN	1.00	R/W	CapNoPush
Read Point:	API status variable	**Write Point :**		Updated on poll

Description: This property is used to control the handling of jam conditions that require a stack of the offending document. If false, the bill acceptor will function normally. If true, the acceptor will enter the STALLED condition when a jam recovery needs to stack. Will throw an exception if not connected, CapNoPush is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Name	Return	Version	Access	When Usable
EscrowOrientation	Orientation	1.00	R	CapOrientation && After DeviceState == Escrow
Read Point:	At bill escrow	**Write Point :**		n/a

Description: This property reflects the orientation of bank notes that are fed into the bill acceptor. Note that bar codes, coupons and bookmarks do not have orientation data. Note that if CapOrientationExt is false then the orientation returned is always ORIENTATION_UNKNOWN.

Name	Return	Version	Access	When Usable
HighSecurity	BOOLEAN	1.00	R/W	Connected
Read Point:	API status variable	**Write Point :**		Updated on poll
Description:	This property is used to control the security criteria applied to the processing of bills. This is a somewhat obsolete concept in that bill acceptance is normally optimize for maximum acceptance and security. Thus most bill acceptors ignore this value.			

Name	Return	Version	Access	When Usable
LastBNFError	BNFErrorStatus	2.80	R	CapBNFStatus
Read Point:	API status variable	**Write Point :**		n/a
Description:	This property is used to report the last BNF status. This property is updated after a call to 'BNFStatus' property.			

Name	Return	Version	Access	When Usable
OrientationCtl	OrientationCtl	1.00	R/W	Connected
Read Point:	API status variable	**Write Point :**		Updated on poll
Description:	This property is used to control the orientation criteria applied to the processing of bills.			

Name	Return	Version	Access	When Usable
OrientationCtlExt	OrientationCtl	1.00	R/W	CapOrientationExt
Read Point:	API status variable	**Write Point :**		Updated on poll
Description:	This property is used to control the orientation criteria applied to the processing of bills. If CapOrientationExt is false then this value is ignored.			

Name	Return	Version	Access	When Usable
VariantID	STRING	1.00	R	CapVariantID
Read Point:	Queries acceptor	**Write Point :**		n/a
Description:	This property string contains the bill variant ID. Will throw an exception if not connected, CapVariantID is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*			

Name	Return	Version	Access	When Usable
VariantNames	STRING[]	1.00	R	Connected
Read Point:	Queries acceptor	**Write Point :**		n/a
Description:	This property is an array of strings that represent the country codes of the currencies accepted by the bill acceptor. Will throw an exception if not connected or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*			

Name	Return	Version	Access	When Usable
VariantPN	STRING	1.00	R	CapVariantPN
Read Point:	Queries acceptor	**Write Point :**		n/a
Description:	This property string contains the bill variant part number. Will throw an exception if not connected, CapVariantPN is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*			

Name	Return	Version	Access	When Usable
APIVersion	STRING	1.00	R	Always
Read Point:	API status variable	**Write Point :**		n/a
Description:	This property contains the version string on the M/POST code.			

3.4 M/POST Document Interface and Classes

MPOST contains several classes that are used to control and report documents.

Actual document classes implement the IDocument Interface and include:

- Bill
- Barcode
- Coupon

3.4.1 IDocument Interface

The IDocument is used as an interface by several different types of documents as show in the next sections. This interface provides a single read-only property. The purpose of the IDocument interface is to define the classes that represent physical documents.

Name	Type	Version	Access	Values
getValueString	STRING	3.50	R	Document value description string

3.4.2 Bill Properties

Name	Type	Version	Access	Values
Country	STRING	1.00	R	Three letter ISO code or “*”.
Value	DOUBLE	1.00	R	
Type	CHAR	1.00	R	‘A’ .. ‘Z’ or ‘*’
Series	CHAR	1.00	R	‘A’ .. ‘Z’ or ‘*’
Compatibility	CHAR	1.00	R	‘A’ .. ‘Z’ or ‘*’
Version	CHAR	1.00	R	‘A’ .. ‘Z’ or ‘*’

3.4.3 Barcode Properties

Name	Type	Version	Access	Values
barcodeValue	STRING	3.50	R	A barcode value string

3.4.4 Coupon Properties

Name	Type	Version	Access	Values
OwnerID	INT32	1.00	R	An integer assigned to the owner of the coupon
Value	DOUBLE	1.00	R	The monetary value of the generic coupon. 0.0D == Free vend coupon 1.0D == $1 coupon 2.0D == $2 coupon 5.0D == $5 coupon

3.5 M/POST Auxiliary Object Properties and Methods

The following section outlines the properties and methods of the additional objects that are made visible to the host application.

3.5.1 M/POST for EBDS AuditContainer Constants

These constants are defined for use with the class's functions. They can act as the orientation value for the following commands { *getRecognizedData, getValidatedData, getStackedData* }.

Name	Type	Version	Access	Value
INT_RIGHT_UP	INT32	2.70	R	0
INT_RIGHT_DOWN	INT32	2.70	R	1
INT_LEFT_UP	INT32	2.70	R	2
INT_LEFT_DOWN	INT32	2.70	R	3

3.5.2 M/POST for EBDS AuditContainer Properties

Name	Type	Version	Access	Values
Maximum NoteCount	INT32	2.70	R	The maximum index value for the contained arrays. This represents the maximum count of unique documents the device can support.

3.5.3 M/POST for EBDS AuditContainer Methods

Return Type	Name	When Usable	Version	Notes
Int[]	getRecognizedData (Orientation orientationValue)	Always	2.70	Parameter is enum value (3.2.5)
Int[]	getRecognizedData (int orientationValue)	Always	2.70	
Int[]	getValidatedData (Orientation orientationValue)	Always	2.70	Parameter is enum value (3.2.5)
Int[]	getRecognizedData (int orientationValue)	Always	2.70	
Int[]	getStackedData (Orientation orientationValue)	Always	2.70	Parameter is enum value (3.2.5)
Int[]	getStackedData (int orientationValue)	Always	2.70	

3.6 M/POST Acceptor Object Methods

3.6.1 Acceptor Object Methods - Overview

Return Type	Name	When Usable	Version
void	Open(String portName)	DeviceState == Disconnected.	2.00
void	Open(String portName, power_up)	DeviceState == Disconnected.	1.00
void	Close(void)	Connected	1.00
void	Calibrate(void)	DeviceState == Idle	1.00
Boolean	CancelAdvancedBookmarkMode(void)	CapAdvBookmark	2.00
Boolean	ClearAudit(void)	CapClearAudit && (DeviceState == Idle || DeviceState == Failed)	2.60
Boolean	DisableCashboxCleanlinessReporting(void)	Connected	2.50
Boolean	EnableCashboxCleanlinessReporting(void)	Connected	2.50
Boolean	EnterAdvancedBookmarkMode(void)	CapAdvBookmark	2.00
void	EscrowReturn(void)	DeviceState == Escrow || DeviceState == PupEscrow	1.00
void	EscrowStack(void)	DeviceState == Escrow || DeviceState == PupEscrow	1.00
void	FlashDownload(STRING file_path)	DeviceState == Idle || DeviceState == DownloadRestart	1.00
void	ClearCashBoxTotal(void)	Connected	1.00
AuditContainer	GetBanknoteAuditData	DeviceState == Idle || DeviceState == Failed	2.70
Long	GetRFIDSerialNumber	Connected && CashboxAttached && DeviceState == (Idle || Jammed || Failed)	3.40
Byte[]	rawTransaction(byte[] command)	Connected	1.00
void	SoftReset(void)	Connected	1.00
void	SetAssetNumber(STRING asset)	Connected && DeviceState == (Idle || Failed)	1.00
void	SetBezel(Bezel b)	Connected	1.10
void	SpecifyEscrowTimeout(INT32 bill_timeout, INT32 barcode_timeout)	Connected	1.00
IDocument (Bill, Barcode or Coupon)	getDocument(void)	After getting Escrowed or Stacked event	3.50

3.6.2 Acceptor Object Methods – Details

Method	Parameters	Return	Version	When Usable
Open	STRING port_name	Void	2.00	When first establishing communication with device

Description: Overloaded method for convenience. It will call the normal 'open' function with a default power up policy ('A').

Method	Parameters	Return	Version	When Usable
Open	STRING port_name PowerUp power_up	Void	1.00	When first establishing communication with device

Description: Connects to a bill acceptor. This function can only be called when first establishing communication with the device or after the Close function has been called. If the Host receives the Disconnect event during normal operations, the API will automatically attempt to re-establish communications without the need to call this function.

If there is an invalid port setting, an exception will be thrown.
If a connection cannot be established, the DISCONNECTED event is sent to the host to indicate a failure.
If the unit is in download mode, the DOWNLOAD_RESTART event is sent to the host.

Method	Parameters	Return	Version	When Usable
Close	Void	Void	1.00	When connected

Description: This function is used to close the connection to the bill acceptor and free up the used port. When completed, this function will raise the DISCONNECTED event.

Recommendation: Set EnableAcceptance property to false prior to calling this function to ensure the unit cannot start processing currency before the connection is closed.

Method	Parameters	Return	Version	When Usable
Calibrate	Void	Void	1.00	DeviceState == Idle

Description: This function becomes empty since V3.50. The calibration feature is deprecated. The following three events are still declared in order not to break the customer application:
CALIBRATE_START
CALIBRATE_PROGRESS
CALIBRATE_FINISH

Method	Parameters	Return	Version	When Usable
CancelAdvanced BookmarkMode	Void	BOOLEAN	2.00	Connected && CapAdvBookmark

Description: Cancels the advanced bookmark mode if the device has been placed in this mode by the EnterAdvancedBookmarkMode function. This will have no effect if the device is not in advanced bookmark mode. It will return TRUE if the device has ACKed the message. This means that the device is NOT in advanced bookmark mode. A return value of FALSE means the device NAKed the message; the device may or may not be in the mode depending on the previous state. Will throw an exception if not connected, CapAdvBookmark is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Method	Parameters	Return	Version	When Usable
Clear Audit	Void	BOOLEAN	2.60	CapClearAudit && (DeviceState == Idle || DeviceState == Failed)

Description: This command requests the device to clear the internal audit counters. The Boolean value that is returned signifies whether or not the device can honor this request; it does not imply that it has performed the operation. Instead, a follow up event will be sent to the host later (~500ms later) which will confirm the request is complete. The event will also inform the host if there was an error that prevented the device from completing the audit clearing. Will throw an exception if not connected or the device state is not *{ Idling, Failed }*

Method	Parameters	Return	Version	When Usable
DisableCashbox CleanlinessReporting	Void	BOOLEAN	2.50	Connected

Description: Disables the reporting of the cashbox cleanliness messages if the device has been placed in this mode by the EnableCashboxCleanlinessReporting function. This will have no effect if the device is not in this mode. It will return TRUE if the device has ACKed the message. This means that the device has successfully disabled the reporting. A return value of FALSE means the device NAKed the message; the device may or may not be in the mode depending on the previous state. Will throw an exception if not connected or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Method	Parameters	Return	Version	When Usable
EnableCashbox CleanlinessReporting	Void	BOOLEAN	2.50	Connected

Description: Enables the reporting of the cashbox cleanliness messages. In this mode, the device will send a status message if a cashbox that requires maintenance is inserted. The function will return TRUE if the device ACKs the message. This means the device will report the cashbox cleanliness. If the return value is FALSE, the device either does not support cashbox cleanliness reporting or could not enter the mode at the time. Will throw an exception if not connected or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Method	Parameters	Return	Version	When Usable
EnterAdvanced BookmarkMode	Void	BOOLEAN	2.00	Connected && CapAdvBookmark

Description: Enters the advanced bookmark mode. In this mode, the device will accept the next document less than 200mm in length. Once a document has been accepted, the device will automatically leave this mode. The function will return TRUE if the device ACKs the message. This means the device is now in the advanced bookmark mode. If the return value is FALSE, the device could not be placed in advanced bookmark mode at this time. Will throw an exception if not connected, CapAdvBookmark is false or the device state is *{ DownloadStart, Downloading, CalibrationStart, Calibrating }*

Method	Parameters	Return	Version	When Usable
EscrowReturn	Void	Void	1.00	DeviceState == Escrow || DeviceState == PupEscrow

Description: Returns the document at escrow. Will throw an exception if not connected.

Method	Parameters	Return	Version	When Usable
EscrowStack	Void	Void	1.00	DeviceState == Escrow || DeviceState == PupEscrow

Description: Stacks the document at escrow. Will throw an exception if not connected.

Method	Parameters	Return	Version	When Usable
FlashDownload	STRING file_path	Void	1.00	DeviceState == Idle || DeviceState == DownloadRestart

Description: This function is used to start the process of updating the flash memory of the bill acceptor. Progress in the download is signaled with the events DOWNLOAD_START, DOWN_LOAD_PROGRESS and DOWNLOAD_FINISH. Will throw an exception if not connected or the device state is not in either Idle or DownloadRestart.

Recommendation: Set EnableAcceptance property to false prior to calling this function to ensure the unit cannot start processing currency.

Method	Parameters	Return	Version	When Usable
ClearCashBoxTotal	Void	Void	1.00	Connected

Description: This function is used to clear the count of bills stored in the cash box. It is recommended to issue this command only when the deviceState is Idle. It has no effect on devices that do not support this command.

Method	Parameters	Return	Version	When Usable
GetBanknote AuditData	Void	Audit Container	2.70	DeviceState == Idle || DeviceState == Failed

Description: This function is used to fetch the audit data from the device. This function can take up to ~20 seconds to complete and it will block all other operations while executing. During this time, note acceptance will be automatically disabled and any note being processed at the time will be returned if it is not in the process of being stacked. Once, complete, the function returns an object that contains 3 core fields (Recognized, Validated, and Stacked). Each field contains four arrays (one array for each orientation { Right Up, Right Down, Left Up, Left Down }) and the length of each array is equal to the maximum number of documents supported. CASHFLOW SC supports 50 documents while SC Advance supports 128 documents. See section 3.5.3 for more details on how to access the data from the AuditContainer object. Will throw an exception if not connected or the device state is not *{ Idling, Failed }*

Method	Parameters	Return	Version	When Usable
GetRFIDSerial Number	Void	Long	3.40	Connected && CashboxAttached && DeviceState == (Idle || Jammed || Failed)

Description: Gets the RFID unique serial number that is associated with the inserted cassette. This functionality requires EASITRAX to be enabled in the bill acceptor's firmware as well as the necessary hardware to support the feature.
Will throw an exception if not connected, cashbox not attached, or the device state is not *{ Idling, Jammed, Failed }.* Will also throw an exception if the read operation fails or if the device firmware does not support the functionality.

Method	Parameters	Return	Version	When Usable
RawTransaction	Byte[] command	Byte[]	1.00	Connected

Description: This function allows the host to issue a custom command. It can be used to send new EBDS messages that have not been integrated within the current M/POST version. Care must be taken when using this feature since it could change the behavior of the device.

Method	Parameters	Return	Version	When Usable
SetAssetNumber	STRING asset	Void	1.00	Connected && DeviceState == (Idle || Failed)

Description: This function is used to set the asset number of the bill acceptor and the cash box. The asset parameter must not be longer than 16 characters or it will throw an exception. Will also throw an exception if the device is not connected or the DeviceState is not either Idle or Failed. This function has no effect on devices when CapAssetNumber is false.

Note: In .NET, this functionality is now duplicated by the AssetNumber property.

Method	Parameters	Return	Version	When Usable
SetBezel	Bezel b	Void	1.00	Connected

Description: This function is used to override the default bezel setting. Note that setting the bezel to the wrong type may result in unpredictable bezel operation. This function has no effect on devices that do not support this feature.

Method	Parameters	Return	Version	When Usable
SoftReset	Void	Void	1.00	When connected

Description: Asks the bill acceptor to perform a reset. This can help recover the device if an abnormal situation is encountered. However, it is not recommended as the first level of correction since it could result in unexpected behavior, especially while processing a note.

Method	Parameters	Return	Version	When Usable
SpecifyEscrowTimeout	INT32 bill_timeout INT32 barcode_timeout	Void	1.00	Connected

Description: This function is used to specify the escrow timeout parameters in use by the bill acceptor. There are two settings, one for bank notes and the other for bar codes. Both are in seconds from 0 through 127 where 0 represents no timeout. This function has no effect on devices that do not support this feature.

3.7 M/POST Acceptor Object Events

This section outlines all of the events that can be raised by the MPOST Acceptor object. This is the only class that will raise events.

Name	Description			
CALIBRATE_FINISH	**Parameter Data :**	none	**Version:**	1.00
	This event signals that calibration has completed. Recommended Action: Host can resume normal operations			
CALIBRATE_PROGRESS	**Parameter Data :**	none	**Version:**	1.00
	Signals that a calibration is in progress. Recommended Action: None			
CALIBRATE_START	**Parameter Data :**	none	**Version:**	1.00
	This message is sent when the calibration process is started and the user needs to insert the calibration document. Recommended Action: Host should enter a special state to handle calibration.			
CASHBOX_CLEANLINESS_DETECTED	**Parameter Data :**	CashBoxCleanlinessEnum Value	**Version:**	2.50
	This event is sent when a cash box is inserted and does not meet the optimal sensor values. In order for this event to be raised, the host must have enabled the feature. See EnableCashboxCleanlinessReporting function. Recommended Action: If value == CleaningRecommended then the host can continue with normal operations; technician should address this cashbox after use. If value == Cleaning Required the device will not be able to enter normal operations. The host should enter Out of Service and request manual intervention. NOTE – This event will only occur when a cashbox is first inserted or directly after the host enables the feature, there is no risk of this event occurring during normal operations.			
CASHBOX_INSTALLED	**Parameter Data :**	none	**Version:**	1.00
	This event is sent when a cash box is inserted into the bill acceptor. Recommended Action: Host can resume normal operations			
CASHBOX_REMOVED	**Parameter Data :**	none	**Version:**	1.00
	This event is sent when the cash box is removed from the bill acceptor. Recommended Action: Host should enter an error condition.			
CHEATED	**Parameter Data :**	none	**Version:**	1.00
	This message is sent when difficulty is encountered transporting a document that may have been caused by a cheat attempt. Recommended Action: Host should enter an error condition.			

<Table Continues on Next Page>

< Continuation from Previous Page - Acceptor Object Event>

Name	Description			
CLEAR_AUDIT_COMPLETE	**Parameter Data :**	BOOLEAN success	**Version:**	2.60
	This event is sent to the host after a request to *Clear Audit* has been made. This event states whether or not the request was successful. It is only sent if the original call to ClearAudit returned 'true', meaning the device acknowledged the request. Recommended Action: a False (failure) signifies a critical error with the device memory. The host should log the event and the device should be analyzed for defects.			
CONNECTED	**Parameter Data :**	none	**Version:**	1.00
	Sent when the bill acceptor can start receiving commands from the host. Recommended Action: Host can enter normal operations as long as the unit is not currently in POWER_UP.			
DISCONNECTED	**Parameter Data :**	none	**Version:**	1.00
	Informs the host that the connection to the bill acceptor has been lost. The API will continue attempt to re-establish communications automatically.			
DOWNLOAD_FINISH	**Parameter Data :**	BOOLEAN success	**Version:**	1.00
	This event indicates that the download has ended. The success parameter is true for success and false for failure. Recommended Action: Host can enter normal operations			
DOWNLOAD_PROGRESS	**Parameter Data :**	INT32 sector_num	**Version:**	1.00
	This event signals that the flash download is ongoing and that sector_num sectors have been sent. Recommended Action: None			
DOWNLOAD_RESTART	**Parameter Data :**	none	**Version:**	1.00
	Informs the host that the bill acceptor has powered up in download mode. Recommended Action: Host should call "flashDownload" function with correct code file			
DOWNLOAD_START	**Parameter Data :**	INT32 total_num	**Version:**	1.00
	This event signals that the download of code has started and that total_num sectors need to be sent to the bill acceptor. Recommended Action: Continue with flash download			
ESCROW	**Parameter Data :**	none	**Version:**	1.00
	Event is sent when a document has reached the escrow position and requires attention from the host. DocType property is now set with correct value. Recommended Action: Host can decide to stack/return the document.			
FAILURE_CLEARED	**Parameter Data :**	none	**Version:**	2.00
	This event is raised when a failure condition has been corrected. Recommended Action: Host can choose to go back in service if able.			
FAILURE_DETECTED	**Parameter Data :**	none	**Version:**	1.00
	This event is raised when a failure condition is detected. Recommended Action: The host should enter an error condition.			

<Table Continues on Next Page>

< Continuation from Previous Page - Acceptor Object Event >

Name	Description			
INVALID_COMMAND	**Parameter Data :**	none	**Version:**	1.00
	This event is raised when the device detects an invalid command. Recommended Action: Log this issue. Investigate cause if many of these issues occur, or it is regularly repeatable.			
JAM_CLEARED	**Parameter Data :**	none	**Version:**	1.00
	This event is sent when the bill acceptor jam is cleared, either automatically or after operator intervention. The error condition is cleared and the application can proceed. Recommended Action: Host can enter normal operations.			
JAM_DETECTED	**Parameter Data :**	none	**Version:**	1.00
	This event is sent when the bill acceptor detects a jam. The bill acceptor requires attention to resolve the jam. Recommended Action: Host should enter an error condition.			
NOTE_RETRIEVED	**Parameter Data :**	none	**Version:**	2.00
	This event is raised when the user has physically removed a rejected/returned note from the mouth of the bill acceptor. This is only applicable on supported units. Recommended Action: No action required. Informational for the host.			
PAUSE_CLEARED	**Parameter Data :**	none	**Version:**	1.00
	This is sent when the bill acceptor bill path is cleared by the customer. Recommended Action: Host can resume normal operations			
PAUSE_DETECTED	**Parameter Data :**	none	**Version:**	1.00
	This is sent when the bill acceptor is paused because the customer is feeding bills too quickly. Recommended Action: Host should enter an error condition.			
POWER_UP	**Parameter Data :**	none	**Version:**	1.00
	Signals when the device has started the power up procedure. The host should not perform any complex actions during this time. Recommended Action: Host should enter a “standby” state and wait for the POWER_UP_COMPLETE event.			
POWER_UP_COMPLETE	**Parameter Data :**	none	**Version:**	2.00
	Signals when the power up procedure has completed. Recommended Action: The host can start/resume normal operations.			
PUP_ESCROW	**Parameter Data :**	none	**Version:**	1.00
	This event is raised when a document is in the escrow position at device startup. This requires attention from the host in order to complete the power up procedure. Recommended Action: Host can decide to stack/return the document.			
REJECTED	**Parameter Data :**	none	**Version:**	1.00
	Informs the host that a document has been rejected. Recommended Action: No action required.			
RETURNED	**Parameter Data :**	none	**Version:**	1.00
	Informs the host when a document has been successfully returned. Recommended Action: No action required.			

<Table Continues on Next Page>

< Continuation from Previous Page - Acceptor Object Event >

Name	Description			
SEND_MESSAGE_FAILURE	**Parameter Data :**	Message message	**Version:**	2.00
	This event is raised when the API fails to send an important message. This applies to most of the function calls that change the state of the device. The message parameter defines the message that failed to be sent. Recommended Action: Host should reattempt the command that failed. If the command continues to fail, the host should enter an error condition.			
STACKED	**Parameter Data :**	none	**Version:**	1.00
	Deprecated (since V3.50). Recommend to use STACKED_WITH_DOC_INFO event below. Sent when a document has been stacked. Recommended Action: Host should issue credit for the stacked document.			
STACKED_WITH_DOC_INFO	**Parameter Data:**	StackedEventArgs	**Version:**	3.50
	A new STACKED event added since V3.50. This new event has Document type (bill, barcode or coupon) and value as the event argument. Any time we need to send a STACKED event, we send this STACKED (with value) at the same time. You only need to handle one of them. And this new event is recommended. Sent when a document has been stacked. Recommended Action: Host should issue credit for the stacked document.			
STACKER_FULL	**Parameter Data :**	none	**Version:**	1.00
	Event is raised when the device can no longer accept notes due to the cashbox becoming full. Recommended Action: Host should enter an error condition.			
STACKER_FULL_CLEARED	**Parameter Data :**	none	**Version:**	2.00
	Signals when a cashbox full event has been cleared. Recommended Action: Host can resume normal operations			
STALL_CLEARED	**Parameter Data :**	none	**Version:**	1.00
	This event is sent when the bill acceptor stall is cleared after operator intervention. The error condition is cleared and the application can proceed. Recommended Action: Host can resume normal operations			
STALL_DETECTED	**Parameter Data :**	none	**Version:**	1.00
	This event is sent when the bill acceptor becomes stalled after an anti-jam effort. The bill acceptor requires attention to resolve the jam. Recommended Action: Host should enter an error condition.			

4 M/POST Functional Operations

The following section outlines what the host application can expect from the API during operations. It divides the functionality into 3 sections. The first covers the operations while not processing any notes (currency). The second describes the operations while accepting notes and finally, the third covers the exceptional situations like errors and jams.

4.1 Non-Document handling

Describes the operations of M/POST when it is not accepting currency.

4.1.1 Connect/Disconnect

The following behavior can be expected when opening and closing a connection to a device.

When connecting to a powered device for the first time, there will be a total of 3 or 4 events sent from the API depending on the device that is connected.

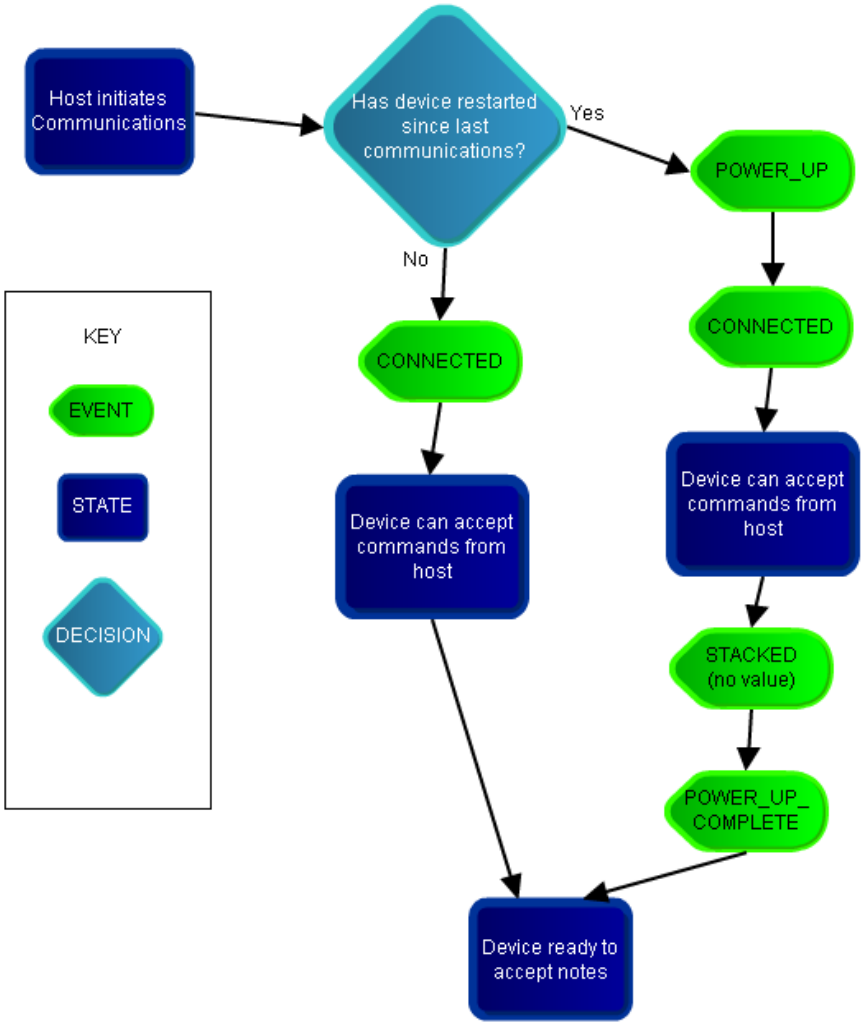
- POWER_UP – This will be the first event. It informs the host that the device is performing its initialization routine.
- CONNECTED – The host is now connected to the device and can perform some operations. The device is still in power up mode and will not be ready to accept currency.
- STACKED – (Deprecated!) Will be a no-value stacked event. This is part of the power up tests that ensure the unit is operating correctly. This message is only sent by SC devices (Including Stackerless) and meant to inform the host that the stacking mechanism is clear and fully functional. The Series 2000 does not send this message.
- STACKED_WITH_DOC_INFO – A new STACKED event with document type and value as event argument. This is part of the power up tests that ensure the unit is operating correctly. This message is only sent by SC devices (Including Stackerless) and meant to inform the host that the stacking mechanism is clear and fully functional. The Series 2000 does not send this message.
- POWER_UP_COMPLETE – The device is now fully operational and ready to accept currency.

Note: If there is no cashbox attached, the API will not report the STACKED event and instead, it will issue the CASHBOX_REMOVED event after the POWER_UP_COMPLETE event.

Please note: Since V3.50, there will be two STACKED events, one without document information, one with Document type and value information as argument. Please handle only ONE and the new event of STACKED_WITH_DOC_INFO is recommended.

If the device has previously communicated with the API without losing power, it will not enter its power up routine and the host can assume full operations once the CONNECTED event is received.

4.1.2 Connect/Disconnect Diagram



4.1.3 Power/Communication Loss

This section outlines the expected behavior of the API when the unit looses power during communications. This assumes the host and API remain active and only the device loses power/communications.

The following assumes the default communications timeout of 30 seconds. See Acceptor Properties – Details for the DisconnectTimeout Property on how to change this value.

If the device does not communicate with the API for a set period of time (30 seconds default), the API will issue a DISCONNECTED event but will continue to attempt to reestablish communications with the device automatically. If the API can re-gain communications, it will receive the following events depending on the reason for the communication loss:

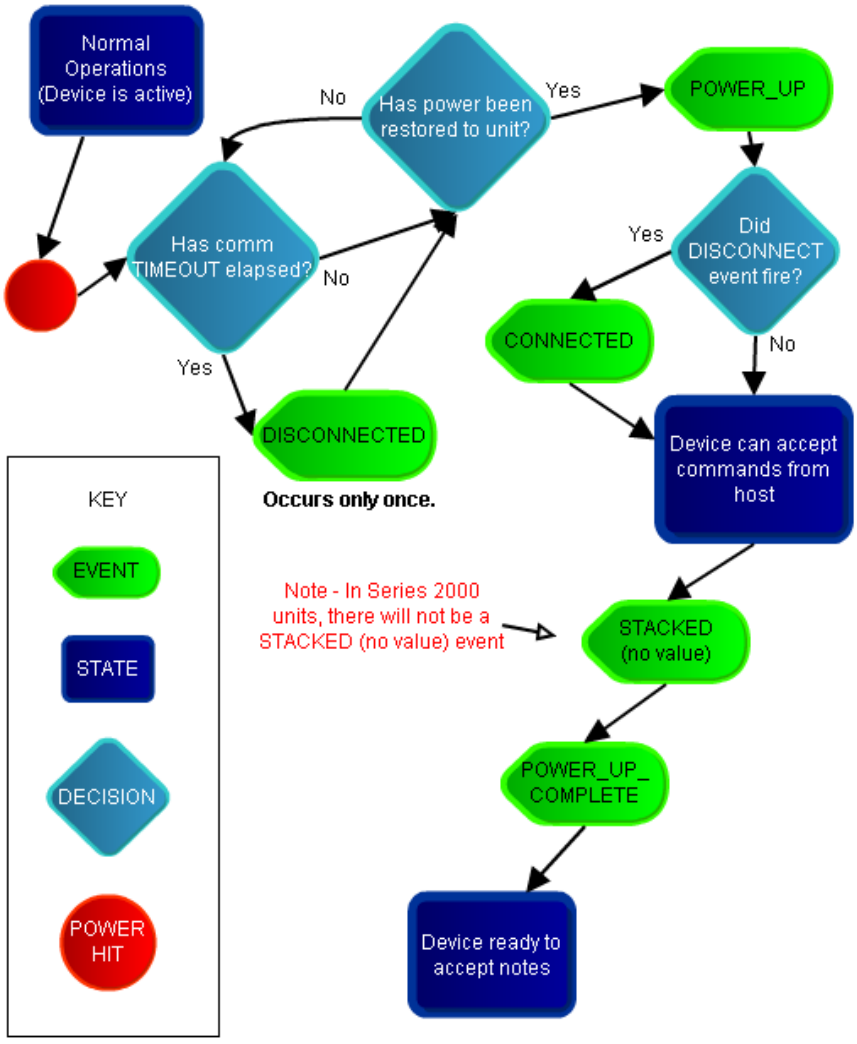
- If the device did not lose power and only communications were disrupted, the host will receive only the CONNECTED event.
- If the device lost power and power is reapplied, the host can expect the 5 events listed (3 for series 2000) for connecting to a device for the first time: POWER_UP, CONNECTED, STACKED no-value (Deprecated. SC devices only), STACKED_WITH_DOC_INFO (New event. SC devices only) and POWER_UP_COMPLETE.

4.1.3.1 Reset during communications:

If the host requested a reset or the power was lost for only a short period of time (less than the disconnected timeout – The DISCONNECTED event was not raised), the unit will only issue the messages related to power up: POWER_UP, STACKED (no doc info), STACKED_WITH_DOC_INFO and POWER_UP_COMPLETE.

Warning – If the disconnect timeout is set too low, the host may receive the DISCONNECTED and CONNECTED event during a reset.

4.1.4 Power/Communication Loss Diagram



4.2 Document handling

This section describes the operations of M/POST while the device is accepting currency.

Note – During any of these scenarios, if the device supports the “Note Retrieved” functionality, the NOTE_RETRIEVED event will be issued once the user removes a note from the mouth of the bill acceptor. This event will occur after the associated REJECTED / RETURNED event. This feature is supported on newer software versions for the SC family of devices.

It is also important to note that if the “AutoStack” property is set, no ESCROW event will be raised. This setting is not recommended if the host intends to save the note values in order to handle power up policies properly.

4.2.1 Terminology Explanation

In many of the diagrams, there is a concept of a “Stacked (no value)” and a “Stacked (with value).” This is a reference to how the actual device reports credit to the MPOST API. The idea is that “value” is in reference to actually currency and the only time a banknote is not reported with its correct value is during some power up conditions. Barcodes will always be referenced as “no value” however; their numeric value can be retrieved from the Barcode property.

4.2.2 Recommended Logic for Handling Notes

The following is a step by step recommendation for general document processing logic that should be implemented by the host application. This logic provides the best way to ensure there are no discrepancies between the host’s count of the notes and the physical count of the money at the end of the day.

- User inserts a document and it reaches escrow
- Host views the DocumentType Property to determine which kind of document it is.
- Host retrieves the correct object based on document type (Bill, Barcode, or Coupon)
- If the host decides to stack this document, the associated value should be stored in non-volatile memory (in case of system wide power loss). This is essential for Pup-A and not as important for Pup-B (See

- Power-Up Handling for further information on the power up policies available)
- Host should wait for the “Stacked” or “Stacked_With_Doc_Info” event.
- Host should confirm the document type and value are the same as when the document was at escrow. (Note that the value can be lost by the device if there was a power loss. This is the importance of the host maintaining the value when the document was at escrow)
- Once this occurs, the host can issue credit for that document and clear the current cache.

NOTE – The MPOST ‘Bill’ Object should only be retrieved immediately after an escrow or stacked event; its value could be unexpectedly reset at any future time.

4.2.3 Normal Operations

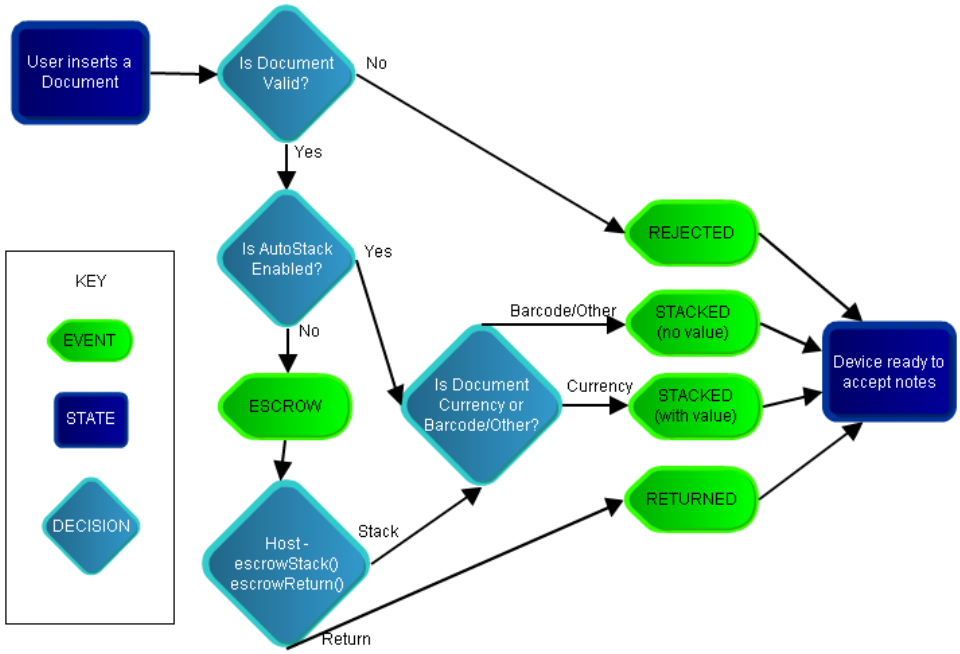
4.2.3.1 Valid currency

If the user inserts a valid note, the API will change its device state from *idling* to *accepting*. Once the note reaches the escrow position, the device will enter the *escrow* state and issue an ESCROW event if the “AutoStack” property is not set. If the “AutoStack” property is set, the API will auto-accept the note, otherwise the API will wait for the host to issue either escrowStack() or escrowReturn().
If the note is to be stacked, the device will enter the *stacking* state. The host can expect the STACKED (with correct value) to be issued once the note is secured. (Warning – with Stackerless units, it is possible for the note to be rejected even after an escrowStack() message is requested by the host). **It is important that the host software only issue credit after the STACKED (or STACKED_WITH_DOC_INFO) event has been received.**
If the host requests the note to be returned, the device will enter the *returning* state and the RETURNED event will be sent from the API once the note has been returned to the user.

4.2.3.2 Invalid Document

If the user inserts an invalid document or the currency is not able to be recognized by the device, the unit will issue a REJECTED event. There will not be an ESCROW event.

4.2.4 Normal Operations - Document Handling Diagram


User inserts a Document
Is Document Valid?
No
Yes
REJECTED
Is AutoStack Enabled?
Yes
No
Barcode/Other
STACKED (no value)
Device ready to accept notes
KEY
EVENT
STATE
DECISION
ESCROW
Is Document Currency or Barcode/Other?
Currency
STACKED (with value)
RETURNED
Host - escrowStack() escrowReturn()
Stack
Return

4.2.5 Power-Up Handling

The following section outlines how the SC devices handle situations where the power is lost while processing a note. Series 2000 units do not follow this convention. It will simply return any document that has not been stacked and it will not issue any rejected event.

The POWER_UP, STACKED (no value), STACKED_WITH_DOC_INFO and POWER_UP_COMPLETE events will be raised during all power up scenarios unless otherwise mentioned. Also, if the power is out for greater than the disconnect timeout, the DISCONNECTED and CONNECTED events will occur as expected. CONNECTED will occur after POWER_UP but before any other event.

If the device loses power while it is accepting a valid currency, the API can act differently depending on the location and status of the note as well as the Power Up Policy that was defined when the connection was established with the API (policy A, B or C)

Series 2000 devices do not support this advanced control over the document at power up. These devices will return the note during any unexpected case involving power hits or extended communications loss.

4.2.5.1 Description of Power Up Policies

The power up policies are settings that govern how the unit will handle exceptional cases during power up. The main purpose of these policies is to ensure notes are handled as expected if the power is lost while processing one. If there is any confusion to these policies, Pup-A is the recommended default policy. The behavior of the policies is explained in more detailed in the following sections, but here a quick overview of the intent behind them:

- Pup-A is designed for the highest “acceptance.” The host needs to maintain the value of the note after the escrow position and the unit will leave it up to the host to accept/reject any document at the escrow position.
- Pup-B and C are designed for higher security. The units will go out of service is the note is at a specified position and will not give the host the chance to process this document.

4.2.5.2 Loss of power while accepting a note (Pre-Escrow)

Device did not reach *escrow* position. If no ESCROW event has occurred, the device will reject the note and issue a REJECTED event. The REJECTED event will occur after the POWER_UP event but before the STACKED (no value) and STACKED_WITH_DOC_INFO events.

4.2.5.3 Loss of power while device has a note in escrow

If the unit was in the *escrow* state (host would receive an ESCROW event if "AutoStack" was set to false) and power was lost before a decision was issued to the device (escrowStack() or escrowReturn()) the API will act according to the power up policy:

- A – The device will issue a PUP_ESCROW event with no value. (It reports "no value" in Policy A. It is up to the host to track the value of the document for the purpose of power up handling) The API will wait for either escrowStack() or escrowReturn(). The autoStack property does not handle this situation so the host must make a call to one of these functions. If the API is told to stack the note, the STACKED and STACKED_WITH_DOC_INFO event will be raised but still contain "no value". If the API is told to return the document, the RETURNED event will fire followed by the STACKED and STACKED_WITH_DOC_INFO (contains no value) as part of the power up routine. Finally, in both situations, the POWER_UP_COMPLETE will be raised once the device is ready to resume normal operations.
- B – The device will issue a FAILURE_DETECTED event. No STACKED and STACKED_WITH_DOC_INF event will fire. This requires an operator to physically remove the note.
- C – Same as policy "B."

4.2.5.4 Loss of power while device is stacking a valid document (Post-Escrow)

This scenario is independent of the "AutoStack" property. It assumes the note has been told to stack. All three policies (A, B and C) have similar behavior but differ slightly in the reporting.

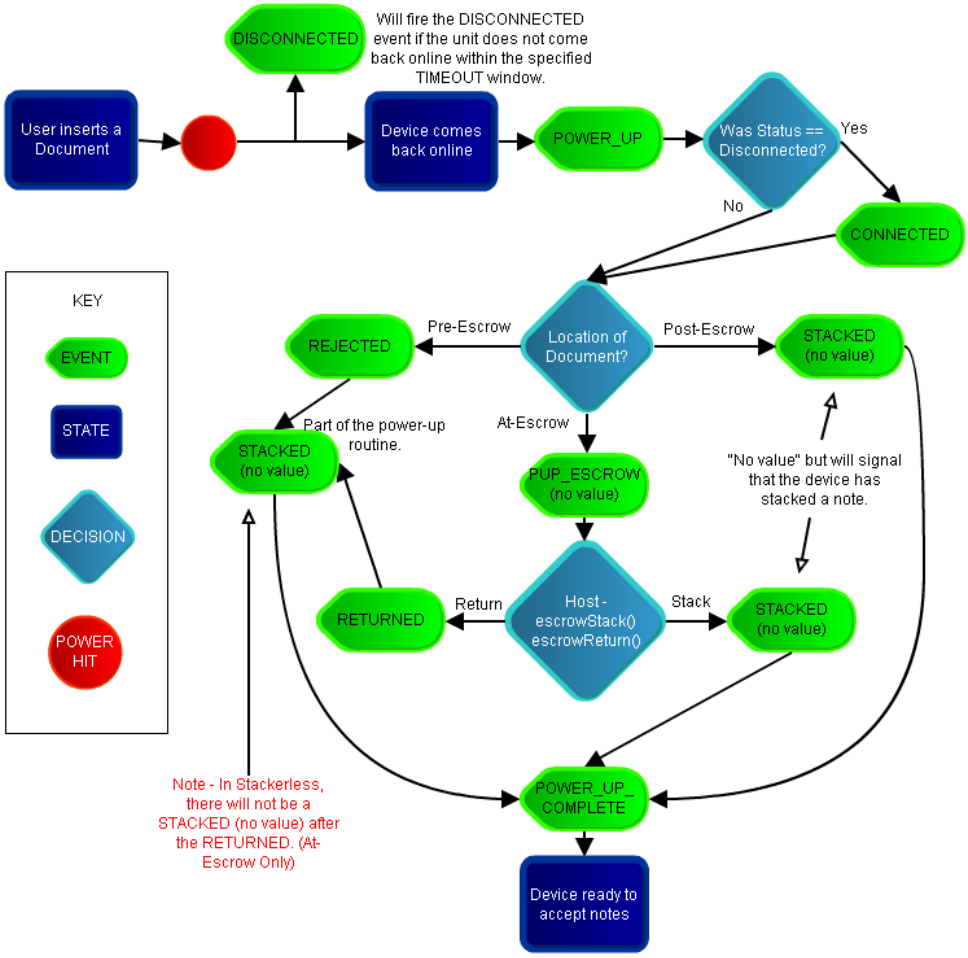
- A – No special events or change in their order. The STACKED events that occur during power up are used to signal when the note has been secured. It will still be reported as "no value" therefore it is best for the host to track the document value when it was at escrow so it can be accounted for properly. (This is why "AutoStack" is not recommended)
- B – Same flow of events however the correct value will be retrievable after the STACKED events.
- C – Same as policy "A."

4.2.5.5 Loss of power while returning a note

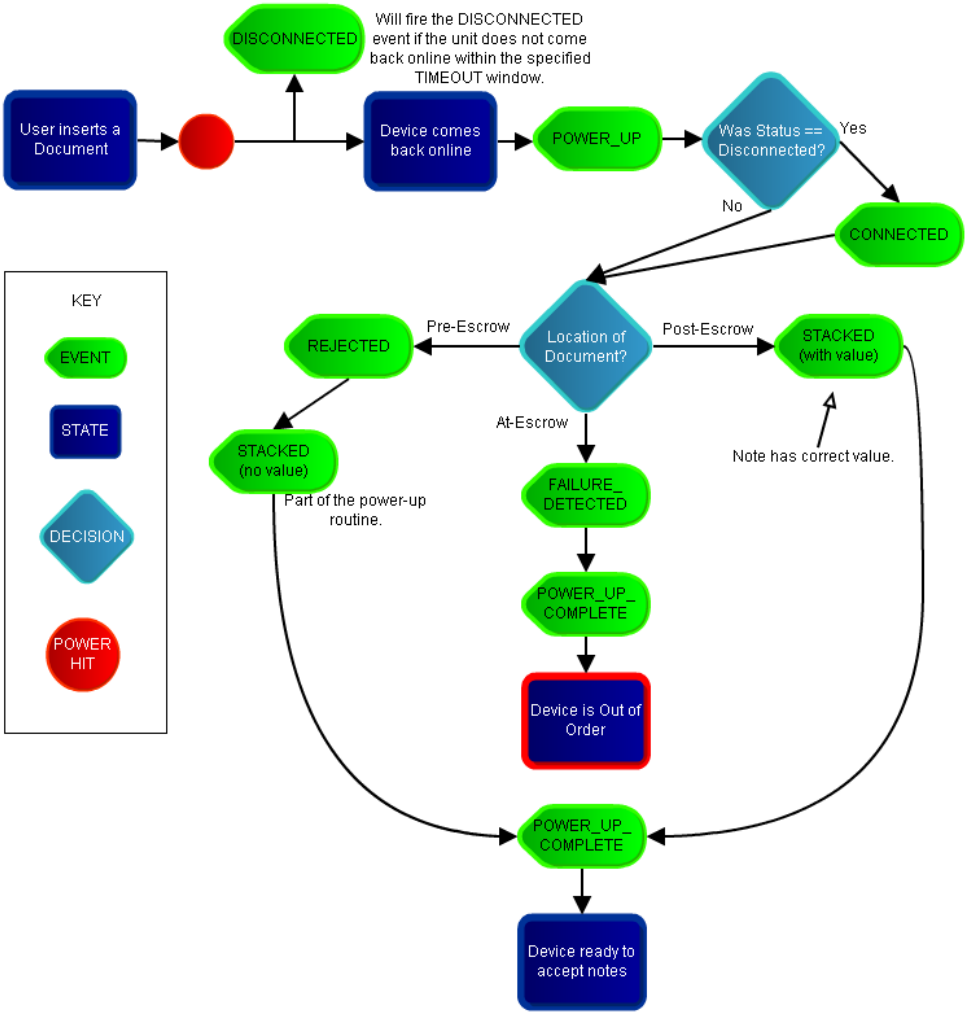
If the device is told the return the note and then the power is lost before the note is returned, the API will issue a REJECTED event in place of the expected RETURNED event and no STACKED will be issued during power up.

4.2.6 Power-Up – Document Handling Diagrams

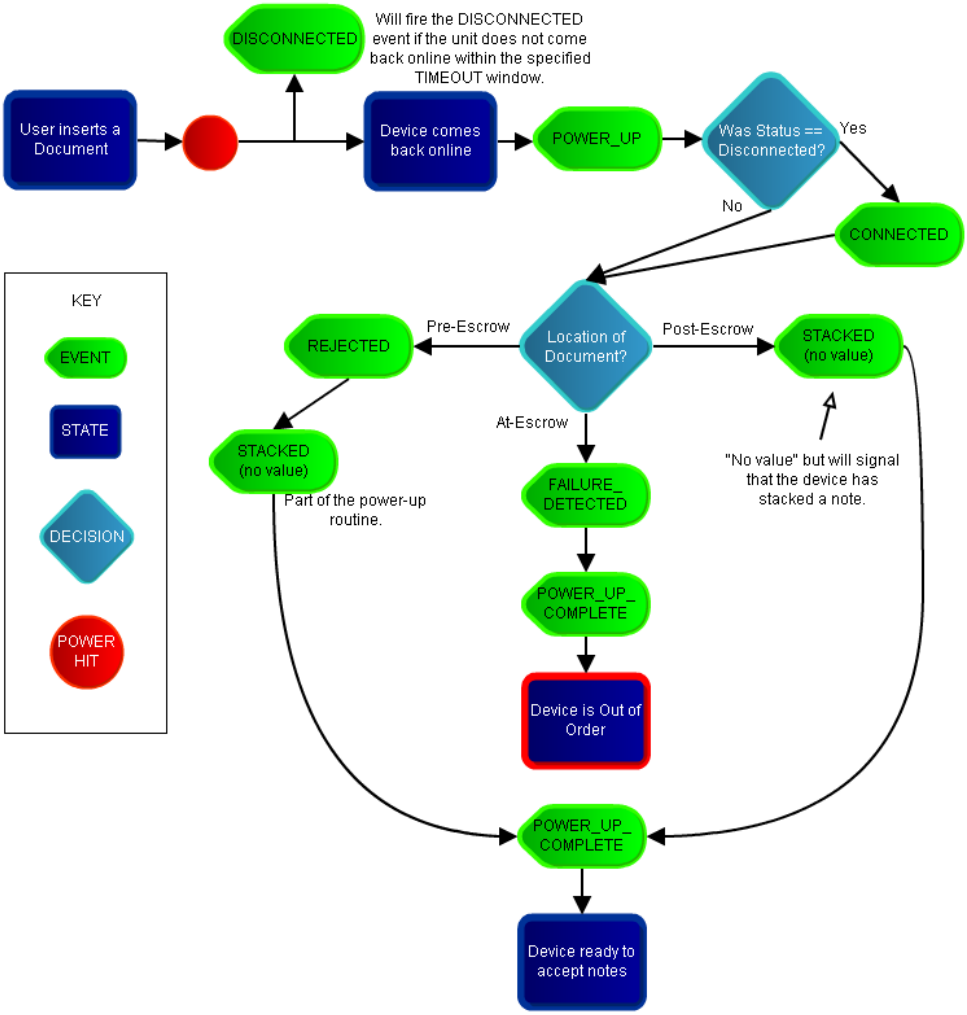
4.2.6.1 Power-Up Policy A



4.2.6.2 Power-Up Policy B



4.2.6.3 Power-Up Policy c



4.2.7 Communications loss during note acceptance

If the communication loss exceeds the disconnect timeout, the DISCONNECTED event will be raised and the CONNECTED event will be raised once communications are re-established. In this situation, the CONNECTED event will always be the first event after a DISCONNECTED event. Although it is not recommended, the following scenarios also apply to situations where the API is told to close/open while processing notes.

If the note has not reached the escrow position, when communications are re-established, the API will issue the ESCROWED event with the correct note value. At this point, it is normal operations. If the "AutoStack" property is set, there will not be an ESCROW event and the API will immediately transition to the *stacking* state.

If communications are lost while the note is in escrow and the API fails to issue the escrowStack() or escrowReturn() function call, the ERROR_ON_SEND_MESSAGE event will be raised. This should signal to the host to repeat the last operation as long as the device is connected. Note – if the "AutoStack" property was set to true and the device failed to send the stack message, it is up to the host to re-try this command.

If communications are lost after the device has been told to stack a valid note, the STACKED message (with correct value) will be issued once communications are re-established. The same behavior can be expected if returning a note: the RETURNED event will occur once communications resume.

Series 2000 devices do not support advanced control over the document after an extended communications loss. These devices will return any note that has not been stacked.

4.3 Exceptional Conditions

4.3.1 Jams

Jams can occur at any time. If the host receives a JAM_DETECTED event, normal operations are suspended in the device. It will no longer accept notes until the jam is cleared. If the jam becomes clear, either due to user interaction or the device self clears on a power up, the JAM_CLEARED event will be raised.

If the device is able to clear the jam by issuing a reset, expect a REJECTED event in addition to the JAM_CLEARED event. The rejected event will not be raised if the user manually clears the jam (provided that the jam is visible by the user and does not require them to remove the bill acceptor).

The JAM_CLEARED event will be raised during power up if the jam requires an operator to visit the machine and manually remove the jam by disconnecting the bill acceptor. Note that this JAM_CLEARED event will only occur on power up if the API has not been shut down as well.

If the jam remains during a power loss or reset, the JAM_DETECTED event will be re-raised during the power up procedure to ensure the host is informed of the status. It will also be resent after the CONNECTED event if communications are lost long enough to trigger the DISCONNECTED event.

4.3.2 Failures

For the most part, failures require an operator to visit the machine in order to correct the problem. A FAILURE_DETECTED event will be raised if the unit enters the failure condition.
If the failure is cleared and the API is not closed, the FAILURE_CLEARED event will be raised.

If the failure remains during a power loss or reset, the FAILURE_DETECTED event will be re-raised during the power up procedure to ensure the host is informed of the status. It will also be resent after the CONNECTED event if communications are lost long enough to trigger the DISCONNECTED event.

4.3.3 Download Restart

A download restart is initiated by the device if a portion of the memory is corrupt. This is usually a result of a power loss during a previous attempt to flash new firmware to the unit. When this occurs, the host shall attempt to flash the last firmware sent to the unit.

4.3.4 Stacker Full and other cashbox related events

A STACKER_FULL event (aka Cashbox full) signals when the device is no longer accepting notes due to the cashbox becoming full. The cashbox must be removed and a new (empty one) replaced. This will cause the CASSETTE_REMOVED and CASSETTE_INSERTED events to fire. Once the new cashbox is inserted, the STACKER_FULL_CLEARED event will fire provided the new cassette is not full. If the new cassette is full, the STACKER_FULL event will be sent again.

If the stacker full status remains during a power loss or reset, the STACKER_FULL event will be re-raised during the power up procedure to ensure the host is informed of the status. It will also be resent after the CONNECTED event if communications are lost long enough to trigger the DISCONNECTED event.

A cassette inserted is the default status of a bill acceptor. If the cassette is ever missing, even on initial communications with the device, the CASSETTE_REMOVED event will be raised.

5 M/POST Integration

The goal of this section is to provide information on getting started with developing using the M/POST Libraries.

5.1 Recommended Key Points

This is a short list of some the more important key points that recommended for the best performance.

- Do not use Auto-Stack. This property is more convenient but opens up the possibility of losing the value of a stacked note when dealing with a power up policy.
- Store data from an escrowed note in non-volatile memory. This way, if there is a system wide power loss, the correct value of a note being stacked can still retrieved by the host. This way there are no miscounts.
- Limit queries to when the device is idle. Some operations may require that the device not be accepting a note at that specific point in time, others may not require this. Either way, it is good practice to only query the device when it is in the IDLE state.
- Do not flood the device with too many synchronous messages. Some messages are specific in nature and force the device to deal with that message in favor of other messages related to its current state. This could result in delayed operations and even prevent the device from completing some operations. (The Series 2000 devices are especially susceptible to this)
- If using a "Session" approach in the host application, turning "EnableAcceptance" off is not instantaneous. The host application must assume that the device could start accepting a note within a short period of time before the next outbound message is sent to the device. The host needs to ensure the device is not processing a note before it closes its session. The "SpecifyEscrowTimeout" function can be beneficial in that it will tell the device to automatically return any notes that may have been inserted when the host has closed the MPOST API.

5.2 Event Thread Safety

Since many of the events are raised by a background thread, it is important to make sure that the thread handling is conducted properly in order to prevent deadlock and inefficient code. As of version 2.30, all events are now completely thread-safe across all platforms. However, there would still be the need to invoke the event thread on the main thread in order to update GUI elements in the host application in a .NET scenario.

5.2.1 .NET

The .NET event handling should make use of the 'InvokeRequired' property of the hosts implementing class. The following code example depicts the proper use of this keyword:

```
private void HandleEscrowedEvent(object sender, EventArgs e)
{
    if (InvokeRequired)
    {
        BeginInvoke(EscrowedDelegate, new object[] { sender, e });
    }
    else
    {
        // Event handling goes here to update the GUI
    }
}
```

5.2.2 Java

As of V2.30, events are completely thread-safe. The host application can safely use the executing thread without the possibility of causing a deadlock situation.