VeriFone Mobile Framework

1.0.4.273

Generated by Doxygen 1.8.6

Mon Dec 15 2014 09:52:00

Contents

1 VeriFone Framework Reference Guide **1**

2 e210/e255/e315 Configuration Parameters **3**

3 e315 Barcode Reference **9**

3.1 Reference . . . 10

3.2 Beeper Tone Definitions . . . 10

3.3 Default Parameter Values . . . 11

3.4 Barcode_host.h reference file . . . 15

4 VCMF API Reference Guide **33**

4.1 Important Security Notice . . . 33

4.1.1 Applicability . . . 34

4.1.2 Distribution and Updates . . . 34

4.1.3 What Does PA-DSS Mean to You? . . . 34

4.1.4 Third Party Applications . . . 35

4.1.5 PA-DSS Guidelines . . . 35

4.1.6 More Information . . . 41

4.2 Introduction . . . 41

4.2.1 Overview . . . 42

4.2.2 Definition of Terms . . . 42

4.2.3 Reference Documents . . . 42

4.3 Installation and Integration Setup . . . 43

4.3.1 System Requirements . . . 43

4.3.2 External Accessory Protocol Strings . . . 43

4.3.3 Setting up PAYware Connect Mobile Integration . . . 43

4.4 VFIConnect Implementation . . . 46

4.4.1 Overview . . . 46

4.4.2 Network Communication . . . 46

4.4.3 Account Registration . . . 47
4.4.4 Transactions . . . 47
4.4.5 Direct XML Implementation . . . 49
4.4.6 NSDictionary Response . . . 50
4.4.7 Hardware Connectivity . . . 51
4.4.8 Track Data . . . 52
4.4.9 Sending Track Data . . . 52
4.5 Unique Serial Number Requirement . . . 53
4.5.1 Generating an unique ID - iOS6 . . . 54
4.5.2 Generating an unique ID - all iOS . . . 54

5 VMF API Reference Guide **57**
5.1 Overall Solution Design . . . 58
5.2 Scope Statement . . . 59
5.3 Application Software Components . . . 60
5.4 Implementation Requirements . . . 61
5.5 Implementation Considerations . . . 62
5.6 Framework Interface . . . 64
5.6.1 XPI Tables . . . 64
5.7 EMV Contact-based Initialization . . . 65
5.8 EMV Contactless Initialization . . . 66
5.9 Financial Transaction Commands . . . 67
5.9.1 EMV Flow by Command . . . 69
5.10 Core Implementation using VMF . . . 70
5.10.1 Integrating with VMF . . . 70
5.10.2 Import the necessary framework/libraries . . . 70
5.10.3 Add Compiler Linker Flags . . . 76
5.10.4 Add Import statements to utilize frameworks . . . 77
5.10.5 Amend the view controller interface to include the framework delegate classes: . . . 77
5.10.6 Implement any/all of the optional delegate protocols used to receive data from VMF: . . . 78
5.10.7 Allocate and initialize the e210/e255/e315 framework objects: . . . 80
5.10.8 Add the e210/e255/e315 device protocol strings to your projects .pList: . . . 80
5.10.9 Sample Project Tutorial . . . 81
5.10.9.1 Step 1: Create New Project . . . 82
5.10.9.2 Step 2: Import Frameworks . . . 84
5.10.9.3 Step 3: Add Compiler Linker Flags . . . 84
5.10.9.4 Step 4: Add Protocol Strings to .pList . . . 85

5.10.9.5 Step 5: Create e210/e255/e315/e335 Singleton Class . . . 85
5.10.9.6 Step 6: Design Interface . . . 89
5.10.9.7 Step 7: Configure Header File . . . 94
5.10.9.8 Step 8: Configure Method File . . . 97
5.11 VMF Best Practices . . . 103
5.11.1 Best Practices Guidelines . . . 103
5.11.2 Establish a Singleton Class to access e210/e255/e315: . . . 103
5.11.3 Monitor the initialized delegates protocols: . . . 104
5.11.4 Delegate Protocol Management: . . . 104
5.11.5 Plan for device resets/modification of settings: . . . 105
5.11.6 Power down scanner on application exit: . . . 105
5.11.7 Power down scanner after sending barcode commands: . . . 105
5.11.8 Let framework manage all connect/disconnect events on background aware app . . . 105
5.11.9 Determine proper barcode scanning mode . . . 106
5.11.10 VMF Threading . . . 107
5.11.11 Keypad Handling . . . 107
5.11.12 MSR Reader . . . 107
5.11.13 Updating e210/e255/e315 Software . . . 108
5.12 Secure Messages for US Commands . . . 108
5.13 PINPad Parameters . . . 114
5.14 EMV Data Formats . . . 116
5.14.1 Master Card Example with Request for Tag Information C36 . . . 116
5.15 General Message Table . . . 117
5.16 Multi-Language Support . . . 120
5.17 Country Specific Implementation Considerations . . . 120
5.18 Troubleshooting . . . 121

6 Deprecated List **123**

7 Hierarchical Index **125**
7.1 Class Hierarchy . . . 125

8 Class Index **127**
8.1 Class List . . . 127

9 File Index **129**
9.1 File List . . . 129

10 Class Documentation **131**

10.1 AsyncSocket Class Reference . . . 131
10.1.1 Method Documentation . . . 134
10.1.1.1 acceptOnInterface:port:error: . . . 134
10.1.1.2 acceptOnPort:error: . . . 134
10.1.1.3 addRunLoopMode: . . . 134
10.1.1.4 canSafelySetDelegate . . . 134
10.1.1.5 connectedAddress . . . 134
10.1.1.6 connectedHost . . . 134
10.1.1.7 connectedPort . . . 134
10.1.1.8 connectToAddress:error: . . . 134
10.1.1.9 connectToAddress:viaInterfaceAddress:withTimeout:error: . . . 134
10.1.1.10 connectToAddress:withTimeout:error: . . . 135
10.1.1.11 connectToHost:onPort:error: . . . 135
10.1.1.12 connectToHost:onPort:withTimeout:error: . . . 135
10.1.1.13 CRData . . . 135
10.1.1.14 CRLFData . . . 135
10.1.1.15 delegate . . . 135
10.1.1.16 description . . . 135
10.1.1.17 disconnect . . . 135
10.1.1.18 disconnectAfterReading . . . 135
10.1.1.19 disconnectAfterReadingAndWriting . . . 135
10.1.1.20 disconnectAfterWriting . . . 136
10.1.1.21 enablePreBuffering . . . 136
10.1.1.22 getCFReadStream . . . 136
10.1.1.23 getCFSocket . . . 136
10.1.1.24 getCFWriteStream . . . 136
10.1.1.25 init . . . 136
10.1.1.26 initWithDelegate: . . . 136
10.1.1.27 initWithDelegate:userData: . . . 136
10.1.1.28 isConnected . . . 136
10.1.1.29 isIPv4 . . . 136
10.1.1.30 isIPv6 . . . 136
10.1.1.31 LFData . . . 136
10.1.1.32 localAddress . . . 136
10.1.1.33 localHost . . . 136
10.1.1.34 localPort . . . 136
10.1.1.35 moveToRunLoop: . . . 136

10.1.1.36 progressOfReadReturningTag:bytesDone:total: . . . 137
10.1.1.37 progressOfWriteReturningTag:bytesDone:total: . . . 137
10.1.1.38 readDataToData:withTimeout:buffer:bufferOffset:maxLength:tag: . . . 137
10.1.1.39 readDataToData:withTimeout:buffer:bufferOffset:tag: . . . 137
10.1.1.40 readDataToData:withTimeout:maxLength:tag: . . . 138
10.1.1.41 readDataToData:withTimeout:tag: . . . 138
10.1.1.42 readDataToLength:withTimeout:buffer:bufferOffset:tag: . . . 138
10.1.1.43 readDataToLength:withTimeout:tag: . . . 139
10.1.1.44 readDataWithTimeout:buffer:bufferOffset:maxLength:tag: . . . 139
10.1.1.45 readDataWithTimeout:buffer:bufferOffset:tag: . . . 139
10.1.1.46 readDataWithTimeout:tag: . . . 139
10.1.1.47 removeRunLoopMode: . . . 139
10.1.1.48 runLoopModes . . . 140
10.1.1.49 setDelegate: . . . 140
10.1.1.50 setRunLoopModes: . . . 140
10.1.1.51 setUserData: . . . 140
10.1.1.52 startTLS: . . . 140
10.1.1.53 unreadData . . . 141
10.1.1.54 userData . . . 141
10.1.1.55 write: . . . 141
10.1.1.56 writeData:withTimeout:tag: . . . 141
10.1.1.57 ZeroData . . . 141
10.1.2 Member Data Documentation . . . 141
10.1.2.1 partialReadBuffer . . . 141
10.1.2.2 theConnectTimer . . . 141
10.1.2.3 theContext . . . 141
10.1.2.4 theCurrentRead . . . 141
10.1.2.5 theCurrentWrite . . . 141
10.1.2.6 theDelegate . . . 141
10.1.2.7 theFlags . . . 141
10.1.2.8 theNativeSocket4 . . . 141
10.1.2.9 theNativeSocket6 . . . 141
10.1.2.10 theReadQueue . . . 141
10.1.2.11 theReadStream . . . 141
10.1.2.12 theReadTimer . . . 141
10.1.2.13 theRunLoop . . . 142
10.1.2.14 theRunLoopModes . . . 142

10.1.2.15 theSocket4 . . . 142
10.1.2.16 theSocket6 . . . 142
10.1.2.17 theSource4 . . . 142
10.1.2.18 theSource6 . . . 142
10.1.2.19 theUserData . . . 142
10.1.2.20 theWriteQueue . . . 142
10.1.2.21 theWriteStream . . . 142
10.1.2.22 theWriteTimer . . . 142
10.2 <AsyncSocketDelegate> Protocol Reference . . . 142
10.2.1 Method Documentation . . . 143
10.2.1.1 onSocket:didAcceptNewSocket: . . . 143
10.2.1.2 onSocket:didConnectToHost:port: . . . 143
10.2.1.3 onSocket:didReadData:withTag: . . . 143
10.2.1.4 onSocket:didReadPartialDataOfLength:tag: . . . 143
10.2.1.5 onSocket:didWriteDataWithTag: . . . 143
10.2.1.6 onSocket:didWritePartialDataOfLength:tag: . . . 143
10.2.1.7 onSocket:shouldTimeoutReadWithTag:elapsed:bytesDone: . . . 144
10.2.1.8 onSocket:shouldTimeoutWriteWithTag:elapsed:bytesDone: . . . 144
10.2.1.9 onSocket:wantsRunLoopForNewSocket: . . . 144
10.2.1.10 onSocket:willDisconnectWithError: . . . 144
10.2.1.11 onSocketDidDisconnect: . . . 144
10.2.1.12 onSocketDidSecure: . . . 144
10.2.1.13 onSocketWillConnect: . . . 145
10.3 CMD_DEF Struct Reference . . . 145
10.3.1 Member Data Documentation . . . 145
10.3.1.1 command . . . 145
10.3.1.2 length . . . 145
10.3.1.3 protoVer . . . 145
10.4 decryptedCredentials Class Reference . . . 145
10.4.1 Detailed Description . . . 146
10.4.2 Method Documentation . . . 146
10.4.2.1 clear . . . 146
10.4.2.2 clear . . . 146
10.4.3 Property Documentation . . . 146
10.4.3.1 clientID . . . 146
10.4.3.2 deviceKey . . . 147
10.4.3.3 deviceType . . . 147

10.4.3.4 serialNumber . . . 147
10.5 HOST_RX_BUF Struct Reference . . . 147
10.5.1 Member Data Documentation . . . 147
10.5.1.1 length . . . 147
10.5.1.2 sequence . . . 147
10.5.1.3 status . . . 147
10.6 HOST_TX_BUF Struct Reference . . . 147
10.6.1 Member Data Documentation . . . 148
10.6.1.1 length . . . 148
10.6.1.2 sequence . . . 148
10.7 ISCP_HIGH Class Reference . . . 148
10.7.1 Detailed Description . . . 148
10.7.2 Method Documentation . . . 148
10.7.2.1 clear . . . 148
10.7.3 Property Documentation . . . 149
10.7.3.1 commandType . . . 149
10.7.3.2 fid . . . 149
10.7.3.3 group . . . 149
10.7.3.4 param . . . 149
10.8 M_BEEP_DEF Struct Reference . . . 149
10.8.1 Member Data Documentation . . . 150
10.8.1.1 duration_1 . . . 150
10.8.1.2 duration_2 . . . 150
10.8.1.3 duration_3 . . . 150
10.8.1.4 duration_4 . . . 150
10.8.1.5 duration_5 . . . 150
10.8.1.6 duration_6 . . . 150
10.8.1.7 freq_1 . . . 150
10.8.1.8 freq_2 . . . 150
10.8.1.9 freq_3 . . . 150
10.8.1.10 freq_4 . . . 150
10.8.1.11 freq_5 . . . 150
10.8.1.12 freq_6 . . . 150
10.8.1.13 pause_1 . . . 150
10.8.1.14 pause_2 . . . 150
10.8.1.15 pause_3 . . . 150
10.8.1.16 pause_4 . . . 150

10.8.1.17 pause_5 . . . 150
10.8.1.18 pause_6 . . . 150
10.9 PARM_DEF Struct Reference . . . 150
10.9.1 Member Data Documentation . . . 151
10.9.1.1 length . . . 151
10.9.1.2 pId . . . 151
10.10 POWER_STATUS Struct Reference . . . 151
10.10.1 Detailed Description . . . 151
10.10.2 Member Data Documentation . . . 151
10.10.2.1 DATA_SYNC_EN . . . 151
10.10.2.2 EXT_PWR . . . 151
10.10.2.3 GANG_CHARGER . . . 152
10.10.2.4 HOST_PWR . . . 152
10.10.2.5 IDEVICE_PRESENT . . . 152
10.10.2.6 POWERSHARE_ON . . . 152
10.11 swipeData Class Reference . . . 152
10.11.1 Detailed Description . . . 153
10.11.2 Method Documentation . . . 153
10.11.2.1 clear . . . 153
10.11.2.2 clear . . . 153
10.11.3 Property Documentation . . . 153
10.11.3.1 cardType . . . 153
10.11.3.2 firstName . . . 153
10.11.3.3 lastName . . . 153
10.11.3.4 maskedPAN . . . 153
10.12 symb_INFO Struct Reference . . . 153
10.12.1 Member Data Documentation . . . 154
10.12.1.1 AIMID . . . 154
10.12.1.2 codeID . . . 154
10.12.1.3 symbology . . . 154
10.13 T_BEEP Struct Reference . . . 154
10.13.1 Member Data Documentation . . . 154
10.13.1.1 dur . . . 154
10.13.1.2 freq . . . 154
10.14 T_BEEP_DEF Struct Reference . . . 154
10.14.1 Member Data Documentation . . . 155
10.14.1.1 b1 . . . 155

10.14.1.2 b2 . . . 155
10.14.1.3 bPause . . . 155
10.15 VFIAccessoryMgr Class Reference . . . 155
10.15.1 Method Documentation . . . 156
10.15.1.1 barcodeNotfication: . . . 156
10.15.1.2 closeAccessory: . . . 156
10.15.1.3 closeDevices . . . 156
10.15.1.4 closeZontalk . . . 156
10.15.1.5 controlNotfication: . . . 156
10.15.1.6 getBarcode . . . 156
10.15.1.7 getControl . . . 156
10.15.1.8 getPrinter . . . 156
10.15.1.9 getSled . . . 156
10.15.1.10 getXPI . . . 156
10.15.1.11 getZontalk . . . 156
10.15.1.12 pinpadNotfication: . . . 156
10.15.1.13 sharedController . . . 156
10.15.2 Member Data Documentation . . . 156
10.15.2.1 _isBTDevice . . . 156
10.15.2.2 _sessions . . . 156
10.15.2.3 _unusedProtocols . . . 156
10.15.3 Property Documentation . . . 156
10.15.3.1 btChannel . . . 156
10.15.3.2 isBTDevice . . . 156
10.15.3.3 postClosing . . . 156
10.15.3.4 sessions . . . 157
10.15.3.5 unusedProtocols . . . 157
10.16 VFIBarcode Class Reference . . . 157
10.16.1 Detailed Description . . . 162
10.16.2 Method Documentation . . . 162
10.16.2.1 abortScan . . . 162
10.16.2.2 activateBarcodeType: . . . 162
10.16.2.3 barcodeTypeEnabled: . . . 162
10.16.2.4 beepOnParsedScan: . . . 162
10.16.2.5 clearDuplicatesBuffer . . . 163
10.16.2.6 closeDevice . . . 163
10.16.2.7 configureBeep: . . . 163

10.16.2.8 consoleEnabled: . . . 163
10.16.2.9 deactivateBarcodeType: . . . 163
10.16.2.10 disableBlocking . . . 164
10.16.2.11 disableProtocol: . . . 164
10.16.2.12 E08_RSA:publicKeyID: . . . 164
10.16.2.13 E08_x509: . . . 164
10.16.2.14 E10: . . . 164
10.16.2.15 E12:p75Data: . . . 164
10.16.2.16 enableBlocking . . . 165
10.16.2.17 enablePKI: . . . 165
10.16.2.18 enableSoftTrigger: . . . 165
10.16.2.19 excludeBarcodeType: . . . 165
10.16.2.20 executeCmd: . . . 165
10.16.2.21 frameworkVersion . . . 166
10.16.2.22 getBeep . . . 166
10.16.2.23 getScanButtonMode . . . 166
10.16.2.24 getVersion . . . 166
10.16.2.25 ignoreDisconnect . . . 166
10.16.2.26 ignoreDuplicates: . . . 166
10.16.2.27 includeAllBarcodeTypes . . . 166
10.16.2.28 includeBarcodeType: . . . 167
10.16.2.29 init . . . 167
10.16.2.30 initDevice . . . 167
10.16.2.31 initDeviceOnServer . . . 167
10.16.2.32 loadKeyInformation_RSA:publicKeyID: . . . 167
10.16.2.33 loadKeyInformation_x509: . . . 168
10.16.2.34 loadPKISig:p75Data: . . . 168
10.16.2.35 logEnabled: . . . 169
10.16.2.36 mAimPattern: . . . 169
10.16.2.37 mAppVersion . . . 169
10.16.2.38 mAutoBeepConfig: . . . 169
10.16.2.39 mAutoBeepConfig:scanBeep:errorBeep: . . . 170
10.16.2.40 mBeepImmediate: . . . 170
10.16.2.41 mButtonStatus . . . 170
10.16.2.42 mContinuousMode: . . . 171
10.16.2.43 mDeviceClose . . . 171
10.16.2.44 mDeviceOpen . . . 171

10.16.2.45mDeviceOpenForce . . . 171
10.16.2.46mDifferentSymbolDecodesTimeout: . . . 171
10.16.2.47mDisableAllSymbologies . . . 171
10.16.2.48mElectronicDisplayRead: . . . 172
10.16.2.49mFirmwareVersion . . . 172
10.16.2.50mGetAutoBeepError . . . 172
10.16.2.51mGetAutoBeepMode . . . 172
10.16.2.52mGetAutoBeepScan . . . 172
10.16.2.53mPicklistMode: . . . 173
10.16.2.54mRestoreDefaults . . . 173
10.16.2.55mScanDataPrefix:value: . . . 173
10.16.2.56mScanDataSuffix1:value: . . . 173
10.16.2.57mScanDataSuffix2:value: . . . 173
10.16.2.58mScanDataTransmitFormat: . . . 174
10.16.2.59mScanTimeout: . . . 174
10.16.2.60mStartScan . . . 174
10.16.2.61mStopScan . . . 174
10.16.2.62mSymbology:data: . . . 174
10.16.2.63mSymbology:value: . . . 175
10.16.2.64mTriggerMode: . . . 175
10.16.2.65mUniqueCodeReport: . . . 175
10.16.2.66processReceivedData: . . . 176
10.16.2.67resetScannerToFactoryDefaults . . . 176
10.16.2.68restartLoopDelay: . . . 176
10.16.2.69scannerFirmwareVersion . . . 176
10.16.2.70sendBeep: . . . 176
10.16.2.71sendCommand: . . . 177
10.16.2.72sendCommand:data: . . . 178
10.16.2.73sendCommand:parameter: . . . 178
10.16.2.74sendCommand:parameter:data: . . . 178
10.16.2.75sendData: . . . 178
10.16.2.76sendISCP: . . . 178
10.16.2.77sendTriggerEvent: . . . 179
10.16.2.78setBeep: . . . 179
10.16.2.79setBeepOff . . . 179
10.16.2.80setBeepOn . . . 179
10.16.2.81setEdge . . . 179

10.16.2.82 setInitLoop: . . . 179
10.16.2.83 setLevel . . . 180
10.16.2.84 setMultiScan . . . 180
10.16.2.85 setPassive . . . 180
10.16.2.86 setScanButtonMode: . . . 180
10.16.2.87 setScanner1D . . . 180
10.16.2.88 setScanner2D . . . 180
10.16.2.89 setScanTimeout: . . . 181
10.16.2.90 setSingleScan . . . 181
10.16.2.91 setSoft . . . 181
10.16.2.92 setTrigger: . . . 181
10.16.2.93 sharedController . . . 181
10.16.2.94 startScan . . . 181
10.16.3 Property Documentation . . . 181
10.16.3.1 barcodeConnected . . . 182
10.16.3.2 barcodeFirmwareRevision . . . 182
10.16.3.3 barcodeHardwareRevision . . . 182
10.16.3.4 barcodeManufacturer . . . 182
10.16.3.5 barcodeModelNumber . . . 182
10.16.3.6 barcodeName . . . 182
10.16.3.7 barcodeSerialNumber . . . 182
10.16.3.8 barcodeVersion . . . 182
10.16.3.9 BTconnected . . . 182
10.16.3.10 connected . . . 182
10.16.3.11 delegate . . . 182
10.16.3.12 eaACC . . . 183
10.16.3.13 firmware . . . 183
10.16.3.14 initialized . . . 183
10.16.3.15 isGen3 . . . 183
10.16.3.16 keyLoadErrorCode . . . 183
10.17 <VFIBarcodeDelegate> Protocol Reference . . . 183
10.17.1 Detailed Description . . . 184
10.17.2 Method Documentation . . . 185
10.17.2.1 barcodeConnected: . . . 185
10.17.2.2 barcodeDataReceived: . . . 186
10.17.2.3 barcodeDataSent: . . . 186
10.17.2.4 barcodeInitialized: . . . 186

10.17.2.5 barcodeLogEntry:withSeverity: . . . 186
10.17.2.6 barcodeMessage: . . . 186
10.17.2.7 barcodeReconnectFinished . . . 187
10.17.2.8 barcodeReconnectStarted . . . 187
10.17.2.9 barcodeScanData: . . . 187
10.17.2.10 barcodeScanData:barcodeType: . . . 187
10.17.2.11 barcodeScanData:code:symbol:aim: . . . 187
10.17.2.12 barcodeSerialData:incoming: . . . 188
10.17.2.13 barcodeTriggerEvent: . . . 188
10.17.2.14 commandResult: . . . 188
10.18 VFIBTBridge Class Reference . . . 188
10.18.1 Method Documentation . . . 189
10.18.1.1 disableBTBridge . . . 189
10.18.1.2 disabledBTBridge . . . 189
10.18.1.3 frameworkVersion . . . 189
10.18.1.4 noDisconnect . . . 189
10.18.1.5 notConnected . . . 189
10.18.1.6 setNoDisconnect: . . . 189
10.19 VFICardData Class Reference . . . 190
10.19.1 Detailed Description . . . 191
10.19.2 Method Documentation . . . 191
10.19.2.1 clear . . . 191
10.19.2.2 sharedController . . . 191
10.19.3 Property Documentation . . . 191
10.19.3.1 accountNumber . . . 191
10.19.3.2 AID . . . 191
10.19.3.3 appLabel . . . 191
10.19.3.4 appPreferredName . . . 191
10.19.3.5 cardHolderName . . . 192
10.19.3.6 emvTags . . . 192
10.19.3.7 entryType . . . 192
10.19.3.8 expiryDate . . . 192
10.19.3.9 isCTLS . . . 192
10.19.3.10 s16track1 . . . 192
10.19.3.11 s16track2 . . . 192
10.19.3.12 serviceCode . . . 192
10.19.3.13 track1 . . . 192

10.19.3.14 track2 . . . 192
10.19.3.15 track3 . . . 192
10.20 VFICipheredData Class Reference . . . 193
10.20.1 Detailed Description . . . 194
10.20.2 Method Documentation . . . 194
10.20.2.1 clear . . . 194
10.20.2.2 sharedController . . . 194
10.20.3 Property Documentation . . . 194
10.20.3.1 dataType . . . 194
10.20.3.2 encryptedBlob_1 . . . 194
10.20.3.3 encryptedBlob_2 . . . 194
10.20.3.4 encryptionType . . . 194
10.20.3.5 EParms . . . 194
10.20.3.6 keyID . . . 194
10.20.3.7 TKG1_Track1 . . . 195
10.20.3.8 TKG1_Track2 . . . 195
10.20.3.9 TKG2_Track1 . . . 195
10.20.3.10 TKG2_Track2 . . . 195
10.20.3.11 TKG3_Track1 . . . 195
10.20.3.12 TKG3_Track2 . . . 195
10.20.3.13 VSD_Blob . . . 195
10.21 VFIControl Class Reference . . . 195
10.21.1 Detailed Description . . . 199
10.21.2 Method Documentation . . . 199
10.21.2.1 closeDevice . . . 199
10.21.2.2 consoleEnabled: . . . 199
10.21.2.3 disableBlocking . . . 199
10.21.2.4 disableKeypadOnUSBPower: . . . 199
10.21.2.5 disableProtocol: . . . 200
10.21.2.6 displayMessage:withX:andY:shouldClearDisplay: . . . 200
10.21.2.7 enableBlocking . . . 200
10.21.2.8 enableLowBatteryNotification: . . . 200
10.21.2.9 enableLowBatteryNotification:withNotificationLevel: . . . 201
10.21.2.10 enableWatchdogTimer:withCountdownValue: . . . 201
10.21.2.11 getBTFirmwareVersion . . . 202
10.21.2.12 getBTFriendlyName . . . 202
10.21.2.13 getBTPIN . . . 202

10.21.2.14 getBTSPPEnable . . . 202
10.21.2.15 getCrashInfo . . . 202
10.21.2.16 getFileHash: . . . 203
10.21.2.17 getGen3DeviceInfo . . . 203
10.21.2.18 getKeypadState . . . 203
10.21.2.19 getKSIVersion . . . 203
10.21.2.20 getLogRecord . . . 204
10.21.2.21 getPowerStatus . . . 204
10.21.2.22 hostPowerEnabled: . . . 204
10.21.2.23 hostPowerEnabled_e335: . . . 204
10.21.2.24 ignoreDisconnect . . . 205
10.21.2.25 init . . . 205
10.21.2.26 initDevice . . . 205
10.21.2.27 initDeviceOnServer . . . 205
10.21.2.28 isHostPowerEnabled . . . 205
10.21.2.29 isHostPowerEnabled_e335 . . . 206
10.21.2.30 keypadBeepEnabled: . . . 206
10.21.2.31 keypadEnabled: . . . 206
10.21.2.32 logEnabled: . . . 206
10.21.2.33 performDisplayDiagnostic . . . 206
10.21.2.34 performKeypadAllKeyDiagnostic . . . 207
10.21.2.35 performMSRDiagnostic . . . 207
10.21.2.36 performPINEntryDiagnostic . . . 207
10.21.2.37 powerDown . . . 207
10.21.2.38 processReceivedData: . . . 207
10.21.2.39 queryBatteryLevel . . . 207
10.21.2.40 queryDeviceLogging . . . 208
10.21.2.41 queryDiagnosticMode . . . 208
10.21.2.42 queryKeypadVersion . . . 208
10.21.2.43 querySoftwareVersion . . . 208
10.21.2.44 rebootDevice . . . 208
10.21.2.45 restartLoopDelay: . . . 208
10.21.2.46 sendCommand: . . . 208
10.21.2.47 sendCommandLRC: . . . 209
10.21.2.48 sendData: . . . 209
10.21.2.49 setContactlessLED: . . . 209
10.21.2.50 setDateAndTime: . . . 209

10.21.2.51 setDeviceLogging: . . . 210
10.21.2.52 setInitLoop: . . . 210
10.21.2.53 sharedController . . . 210
10.21.2.54 startDiagnosticMode . . . 210
10.21.2.55 stopDiagnosticMode . . . 210
10.21.3 Property Documentation . . . 211
10.21.3.1 batteryLevel . . . 211
10.21.3.2 BTconnected . . . 211
10.21.3.3 connected . . . 211
10.21.3.4 controlConnected . . . 211
10.21.3.5 controlFirmwareRevision . . . 211
10.21.3.6 controlHardwareRevision . . . 211
10.21.3.7 controlManufacturer . . . 211
10.21.3.8 controlModelNumber . . . 211
10.21.3.9 controlName . . . 211
10.21.3.10 controlSerialNumber . . . 211
10.21.3.11 delegate . . . 211
10.21.3.12 eaACC . . . 212
10.21.3.13 initialized . . . 212
10.21.3.14 isGen3 . . . 212
10.21.3.15 vfiGen3DeviceInfo . . . 212
10.21.3.16 vfiKeypadVersion . . . 212
10.21.3.17 vfiSoftwareVersion . . . 212
10.22 <VFIControlDelegate> Protocol Reference . . . 212
10.22.1 Detailed Description . . . 213
10.22.2 Method Documentation . . . 213
10.22.2.1 controlBatteryEvent:batteryEvent: . . . 213
10.22.2.2 controlBatteryLevel: . . . 213
10.22.2.3 controlConnected: . . . 214
10.22.2.4 controlDataReceived: . . . 214
10.22.2.5 controlDataSent: . . . 214
10.22.2.6 controlInitialized: . . . 214
10.22.2.7 controlKeypadVersion: . . . 215
10.22.2.8 controlLogEntry:withSeverity: . . . 216
10.22.2.9 controlReconnectFinished . . . 216
10.22.2.10 controlReconnectStarted . . . 216
10.22.2.11 controlSerialData:incoming: . . . 216

10.22.2.12controlSoftwareVersion: . . . 216
10.23 VFIDataEntry Class Reference . . . 216
10.23.1 Detailed Description . . . 217
10.23.2 Method Documentation . . . 218
10.23.2.1 clear . . . 218
10.23.2.2 sharedController . . . 218
10.23.3 Property Documentation . . . 218
10.23.3.1 accountSelected . . . 218
10.23.3.2 amountConfirmed . . . 218
10.23.3.3 cashbackAmount . . . 218
10.23.3.4 cashBackSelected . . . 218
10.23.3.5 pinBlock . . . 218
10.23.3.6 serialNumber . . . 218
10.23.3.7 surchargeIndicator . . . 218
10.23.3.8 tipAmount . . . 218
10.23.3.9 tipSelected . . . 219
10.24 VFIDiagnostics Class Reference . . . 219
10.24.1 Detailed Description . . . 220
10.24.2 Method Documentation . . . 220
10.24.2.1 clear . . . 220
10.24.2.2 sharedController . . . 220
10.24.3 Property Documentation . . . 220
10.24.3.1 ctlsReaderFirmwareVersion . . . 220
10.24.3.2 emvVersion . . . 220
10.24.3.3 osversion . . . 220
10.24.3.4 pinpadSerialNumber . . . 220
10.24.3.5 svcInfoPartNumber . . . 220
10.24.3.6 svcInfoSerialNumber . . . 220
10.24.3.7 vspVersion . . . 221
10.24.3.8 vxciVersion . . . 221
10.24.3.9 xpiVersion . . . 221
10.25 VFIEMAFiles Class Reference . . . 221
10.25.1 Detailed Description . . . 222
10.25.2 Method Documentation . . . 222
10.25.2.1 clear . . . 222
10.25.2.2 sharedController . . . 222
10.25.3 Property Documentation . . . 222

10.25.3.1 ALLOW_MESSAGES . . . 222
10.25.3.2 DOWNLOAD_ON_TIMEOUT . . . 222
10.25.3.3 FileName . . . 222
10.25.3.4 GID . . . 222
10.25.3.5 Logical . . . 223
10.25.3.6 NAME . . . 223
10.25.3.7 VERSION . . . 223
10.26 VFIEMVAuthorization Class Reference . . . 223
10.26.1 Detailed Description . . . 224
10.26.2 Method Documentation . . . 224
10.26.2.1 clear . . . 224
10.26.2.2 sharedController . . . 224
10.26.3 Property Documentation . . . 224
10.26.3.1 emvTags . . . 224
10.26.3.2 pinSuccessful . . . 224
10.27 VFIEMVCompletionData Class Reference . . . 224
10.27.1 Detailed Description . . . 225
10.27.2 Method Documentation . . . 225
10.27.2.1 clear . . . 225
10.27.2.2 sharedController . . . 225
10.27.3 Property Documentation . . . 225
10.27.3.1 cardRemovalResult . . . 225
10.27.3.2 emvTags . . . 226
10.27.3.3 validateMacResult . . . 226
10.27.3.4 writeToScriptResult . . . 226
10.28 VFIEMVConfiguration Class Reference . . . 227
10.28.1 Detailed Description . . . 229
10.28.2 Method Documentation . . . 229
10.28.2.1 clear . . . 229
10.28.3 Property Documentation . . . 229
10.28.3.1 acquireID . . . 229
10.28.3.2 additionalCapacity . . . 229
10.28.3.3 autoSelectApplication . . . 229
10.28.3.4 blackListCardSupportFlag . . . 229
10.28.3.5 capkIndex . . . 230
10.28.3.6 cardStatusDisplay . . . 230
10.28.3.7 countryCode . . . 230

10.28.3.8 currencyCode . . . 230
10.28.3.9 defaultDDOL . . . 230
10.28.3.10 defaultTDOL . . . 230
10.28.3.11 fallbackAllowedFlag . . . 230
10.28.3.12 floorLimit . . . 230
10.28.3.13 issAcqFlag . . . 230
10.28.3.14 issuerReference . . . 230
10.28.3.15 maxTargetRSPercent . . . 230
10.28.3.16 merchantCategoryCode . . . 231
10.28.3.17 merchantForceOnlineFlag . . . 231
10.28.3.18 merchantID . . . 231
10.28.3.19 modifyCandidateListFlag . . . 231
10.28.3.20 noDisplaySupportFlag . . . 231
10.28.3.21 pinBypassFlag . . . 231
10.28.3.22 pinDevriKey . . . 231
10.28.3.23 pinDevriMacro . . . 231
10.28.3.24 pinFormat . . . 231
10.28.3.25 pinMacroNumber . . . 231
10.28.3.26 pinScriptNumber . . . 231
10.28.3.27 pinTimeout . . . 232
10.28.3.28 rfu1String . . . 232
10.28.3.29 rfu2String . . . 232
10.28.3.30 rfu3String . . . 232
10.28.3.31 rsThreshold . . . 232
10.28.3.32 schemeReference . . . 232
10.28.3.33 tacDefault . . . 232
10.28.3.34 tacDenial . . . 232
10.28.3.35 tacOnline . . . 232
10.28.3.36 targetRSPercent . . . 232
10.28.3.37 termCurExp . . . 232
10.28.3.38 terminalCapacity . . . 233
10.28.3.39 terminalCategoryCode . . . 233
10.28.3.40 terminalID . . . 233
10.28.3.41 terminalType . . . 233
10.28.3.42 trmDataPresent . . . 233
10.28.3.43 vfi_rfu1 . . . 233
10.28.3.44 vfi_rfu2 . . . 233

10.28.3.45 vfi_rfu3 . . . 233
10.29 VFIEMVResponse Class Reference . . . 233
10.29.1 Detailed Description . . . 234
10.29.2 Method Documentation . . . 234
10.29.2.1 clear . . . 234
10.29.3 Property Documentation . . . 235
10.29.3.1 clearBeforeScriptWrite . . . 235
10.29.3.2 displayResult . . . 235
10.29.3.3 hostDecision . . . 235
10.29.3.4 hostTagString . . . 235
10.29.3.5 macBlock . . . 235
10.29.3.6 macData . . . 235
10.29.3.7 numberOfScripts . . . 235
10.29.3.8 scriptData . . . 235
10.29.3.9 scriptId . . . 235
10.29.3.10 scriptLength . . . 235
10.29.3.11 takKey . . . 236
10.29.3.12 pkKey . . . 236
10.30 VFIEMVTags Class Reference . . . 236
10.30.1 Detailed Description . . . 236
10.30.2 Method Documentation . . . 237
10.30.2.1 clear . . . 237
10.30.2.2 sharedController . . . 237
10.30.3 Property Documentation . . . 237
10.30.3.1 emvTags . . . 237
10.31 VFIEncryptedBarcode Class Reference . . . 237
10.31.1 Detailed Description . . . 238
10.31.2 Method Documentation . . . 238
10.31.2.1 clear . . . 238
10.31.2.2 sharedController . . . 238
10.31.3 Property Documentation . . . 238
10.31.3.1 decryptedLength . . . 238
10.31.3.2 encryptedData . . . 238
10.31.3.3 publicKey . . . 238
10.31.3.4 scanIndex . . . 238
10.31.3.5 terminalSerialNumber . . . 238
10.32 VFIEncryptionData Class Reference . . . 239

10.32.1 Detailed Description . . . 239
10.32.2 Method Documentation . . . 239
10.32.2.1 clear . . . 239
10.32.2.2 sharedController . . . 240
10.32.3 Property Documentation . . . 240
10.32.3.1 accountSelection . . . 240
10.32.3.2 macBlock . . . 240
10.32.3.3 pinBlock . . . 240
10.32.3.4 pinBlockStr . . . 240
10.32.3.5 serialNumber . . . 240
10.33 VFIFinancialData Class Reference . . . 240
10.33.1 Detailed Description . . . 241
10.33.2 Method Documentation . . . 241
10.33.2.1 clear . . . 241
10.33.3 Property Documentation . . . 241
10.33.3.1 additionalPromptResponses . . . 241
10.33.3.2 additionalPrompts . . . 241
10.33.3.3 cardData . . . 241
10.33.3.4 cardholderEntries . . . 241
10.33.3.5 emvAuthorization . . . 242
10.33.3.6 emvTags . . . 242
10.33.3.7 encryptedData . . . 242
10.34 VFIGen3DeviceInfo Class Reference . . . 242
10.34.1 Detailed Description . . . 243
10.34.2 Method Documentation . . . 243
10.34.2.1 clear . . . 243
10.34.2.2 sharedController . . . 243
10.34.3 Property Documentation . . . 243
10.34.3.1 AMMVersion . . . 243
10.34.3.2 batteryChargerStatus . . . 243
10.34.3.3 batteryCurrent . . . 244
10.34.3.4 batteryFullCharge . . . 244
10.34.3.5 batteryRemainingCharge . . . 244
10.34.3.6 batteryTemperature . . . 244
10.34.3.7 bundleVersion . . . 244
10.34.3.8 hardwareVersion . . . 244
10.34.3.9 modelNumber . . . 244

10.34.3.10 partNumber . . . 244
10.34.3.11 serialNumber . . . 244
10.34.3.12 SSLCLIVersion . . . 244
10.34.3.13 XPIVersion . . . 244
10.35 VFIKeypadVersion Class Reference . . . 245
10.35.1 Detailed Description . . . 245
10.35.2 Method Documentation . . . 245
10.35.2.1 clear . . . 245
10.35.3 Property Documentation . . . 245
10.35.3.1 BootloaderMajor . . . 245
10.35.3.2 BootloaderMinor . . . 246
10.35.3.3 FirmwareMajor . . . 246
10.35.3.4 FirmwareMinor . . . 246
10.35.3.5 Processor . . . 246
10.36 VFIPinpad Class Reference . . . 246
10.36.1 Detailed Description . . . 258
10.36.2 Method Documentation . . . 258
10.36.2.1 acceptEncryptPIN: . . . 258
10.36.2.2 acceptEncryptPINMessage:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processingMessage: . . . 259
10.36.2.3 amountEntry:message1:message2: . . . 259
10.36.2.4 amountIntroduction:amountFormat:message1:message2: . . . 260
10.36.2.5 C18:threshold:target:targetPercent: . . . 261
10.36.2.6 C19:value: . . . 261
10.36.2.7 C20: . . . 261
10.36.2.8 C21: . . . 261
10.36.2.9 C25:clear:numScripts:length:data: . . . 262
10.36.2.10 C30:language:amount:otherAmount: . . . 262
10.36.2.11 C30:language:amount:otherAmount:isRefundCTLS:disableCTLS: . . . 262
10.36.2.12 C32:otherAmount:merchantDecision:minRequestObjects: . . . 262
10.36.2.13 C32:otherAmount:merchantDecision:minRequestObjects:displayResult: . . . 262
10.36.2.14 C32:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount: . . . 262
10.36.2.15 C32:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:displayAppExpired: . . . 263
10.36.2.16 C34:displayResult:hostTags: . . . 263
10.36.2.17 C36: . . . 263
10.36.2.18 C52:otherAmount:merchantDecision:minRequestObjects: . . . 263

10.36.2.19 C54:displayResult:hostTags: . . . 263
10.36.2.20 cancelCommand . . . 264
10.36.2.21 cancelCommandWithDisplayOptions: . . . 264
10.36.2.22 cancelCommandWithDisplayOptions:andEMVFallback:disableKeyboard: . . . 264
10.36.2.23 cancelPendingCommand . . . 265
10.36.2.24 cancelPendingCommandNoData . . . 265
10.36.2.25 cardAuthorization:otherAmount:merchantDecision:minRequestObjects: . . . 265
10.36.2.26 cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult: . . . 266
10.36.2.27 cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount: . . . 269
10.36.2.28 cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:displayAppExpired: . . . 270
10.36.2.29 clearAllData . . . 272
10.36.2.30 clearCAPK . . . 272
10.36.2.31 closeDevice . . . 273
10.36.2.32 combinedPromptCommand:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBackOptions:surchargeOptions: . . . 273
10.36.2.33 completeOnlineEMV:displayResult:hostTags: . . . 277
10.36.2.34 consoleEnabled: . . . 279
10.36.2.35 copyCAPKFile . . . 279
10.36.2.36 copyLastEMVTags . . . 279
10.36.2.37 copyPromptData . . . 280
10.36.2.38 copyStringResponse . . . 280
10.36.2.39 createCAPK:pkIndex:modulus:exponent:checksum:expiryDate: . . . 280
10.36.2.40 createCAPKFile:pkIndex:modulus:exponent:checksum:expiryDate: . . . 281
10.36.2.41 createMac: . . . 282
10.36.2.42 createTable:numRecords: . . . 282
10.36.2.43 ctlsOPTFLAG . . . 284
10.36.2.44 ctlsState . . . 284
10.36.2.45 D01: . . . 285
10.36.2.46 D11:pkIndex:modulus:exponent:checksum:expiryDate: . . . 285
10.36.2.47 D11a:pkIndex:modulus:exponent:checksum:expiryDate: . . . 285
10.36.2.48 D12 . . . 285
10.36.2.49 D13:AID:partialNameMatch:supportedAID:termAVN:term2ndAVN:recommendedAIDName: . . . 286
10.36.2.50 D14:tableData: . . . 286
10.36.2.51 D15:numberTrans:scheme:regId:CSNList:emvTableRecord:CSNFile: . . . 286

10.36.2.52 D16:numRecords: . . . 286
10.36.2.53 D17:emvRecord:AID:maxAIDLen:appFlow:partialName:acctSelect:enableCtls:transScheme: . . . 286
10.36.2.54 D18:sig:CVM: . . . 286
10.36.2.55 D19:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ: . . . 287
10.36.2.56 D19:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities: . . . 287
10.36.2.57 D19:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:ctlsCountry:ctlsCurrency: . . . 287
10.36.2.58 D20:ctlsMerchantType:ctlsTermTransInfo:ctlsTermTransType:ctlsReqReceiptLimit:ctlsOptionStatus:ctlsReaderFlooorLimit: . . . 287
10.36.2.59 D25:ctlsMode:pdolSupport: . . . 287
10.36.2.60 D25:ctlsMode:pdolSupport:valueLinkCtls:amexMode: . . . 288
10.36.2.61 D26:aid:version: . . . 288
10.36.2.62 D27 . . . 288
10.36.2.63 D30:line2:line3:line4:amt1:amt2:amt3:amt4: . . . 288
10.36.2.64 dataEntryRequest:language:tip:cashback:surcharge: . . . 288
10.36.2.65 dataEntryRequestPrompt: . . . 289
10.36.2.66 derivedKey: . . . 290
10.36.2.67 diagnosticInfo . . . 290
10.36.2.68 diagnosticInfoExtended . . . 291
10.36.2.69 disableBlocking . . . 291
10.36.2.70 disableMSR . . . 291
10.36.2.71 displayMessage:clearScreen: . . . 291
10.36.2.72 displayMessageAndWait:line2: . . . 291
10.36.2.73 displayMessageAndWait:line2:line3:line4: . . . 292
10.36.2.74 displayMessagePromptInput:echo:timeout:maxLen:entryType: . . . 293
10.36.2.75 displayMessages:Line2:Line3:Line4: . . . 293
10.36.2.76 displayMessages:Line2:Line3:Line4:timeout: . . . 294
10.36.2.77 displayMessages:message1:message2:clearScreen: . . . 294
10.36.2.78 doInteracRequest:language:amount:pan:mac: . . . 295
10.36.2.79 E00 . . . 296
10.36.2.80 E02 . . . 296
10.36.2.81 E04 . . . 296
10.36.2.82 E06 . . . 296
10.36.2.83 E08_RSA:publicKeyID: . . . 297
10.36.2.84 E08_x509: . . . 297

10.36.2.85 E10: . . . 297
10.36.2.86 E10:keyIndex: . . . 297
10.36.2.87 E12:p75Data: . . . 297
10.36.2.88 E14 . . . 297
10.36.2.89 E20: . . . 298
10.36.2.90 enableBlocking . . . 298
10.36.2.91 enableMSR . . . 298
10.36.2.92 enableMSRDualTrack . . . 298
10.36.2.93 frameworkVersion . . . 298
10.36.2.94 generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry: . . . 298
10.36.2.95 generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry-:displayCanceled:restoreScreen: . . . 299
10.36.2.96 generalPromptDefault:message2:minLength:maxLength:language:initialValue:allowSkip: . . . 300
10.36.2.97 getCardData:language:amount:otherAmount: . . . 301
10.36.2.98 getCardData:language:amount:otherAmount:isRefundCTLS:disableCTLS: . . . 302
10.36.2.99 getCipheredData . . . 303
10.36.2.100 getEMVTag: . . . 305
10.36.2.101 getEMVTags: . . . 305
10.36.2.102 getEncryptionMode . . . 306
10.36.2.103 getKeyID . . . 306
10.36.2.104 getOSVersion . . . 307
10.36.2.105 getPKICipheredData . . . 307
10.36.2.106 getXPIVersion . . . 307
10.36.2.107 I72 . . . 307
10.36.2.108 ignoreDisconnect . . . 307
10.36.2.109 init . . . 307
10.36.2.110 initDevice . . . 308
10.36.2.111 initDeviceOnServer:port: . . . 308
10.36.2.112 initializeContactless:sig:CVM: . . . 308
10.36.2.113 initializeIdle . . . 309
10.36.2.114 initServerOnPort: . . . 309
10.36.2.115 isGen3Server . . . 309
10.36.2.116 loadKeyInformation_RSA:publicKeyID: . . . 309
10.36.2.117 loadKeyInformation_x509: . . . 310
10.36.2.118 loadPKISig:p75Data: . . . 310
10.36.2.119 logEnabled: . . . 311

10.36.2.120 menuChoiceSelection:message2:choice1:choice2:choice3:choice4: 311
10.36.2.121 ioResponseNextCommand 312
10.36.2.122 obtainCardData: 312
10.36.2.123 obtainCardData:displayIdle: 313
10.36.2.124 obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode: 314
10.36.2.125 obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting-:masking:timeout: 315
10.36.2.126 obtainTrack2Data:optionalAmount: 316
10.36.2.127 performTGKUpdateRequest 317
10.36.2.128 pinBypass: 317
10.36.2.129 pinManagementCardAuthorization:otherAmount:merchantDecision:minRequest-Objects: 318
10.36.2.130 pinManagementCompleteOnlineEMV:displayResult:hostTags: 319
10.36.2.131 processReceivedData: 321
10.36.2.132 PS01: 321
10.36.2.133 PS02: 321
10.36.2.134 Q40 321
10.36.2.135 Q41 322
10.36.2.136 Q42 322
10.36.2.137 removeCard 322
10.36.2.138 requestEMAFiles 322
10.36.2.139 requestPINRentry: 323
10.36.2.140 requestPINRentry:pinpadPrompt:timeout: 323
10.36.2.141 requestStringInput:timeout:maxLen: 324
10.36.2.142 requestStringInput:timeout:maxLen:entryType: 324
10.36.2.143 restartLoopDelay: 325
10.36.2.144 retrieveEParmsData 325
10.36.2.145 S00 327
10.36.2.146 S01 327
10.36.2.147 S07:pin:balance: 327
10.36.2.148 S11:line2: 327
10.36.2.149 S11:line2:line3:line4: 327
10.36.2.150 S12:message2:minLength:maxLength:language:allowSkip:secureEntry: 328
10.36.2.151 S12:message2:minLength:maxLength:language:allowSkip:secureEntry:display-Canceled:restoreScreen: 328
10.36.2.152 S12Default:message2:minLength:maxLength:language:initialValue:allowSkip: 328
10.36.2.153 S13:message2:choice1:choice2:choice3:choice4: 328
10.36.2.154 S14:Line2:Line3:Line4: 328

10.36.2.155 514:Line2:Line3:Line4:timeout: . . . 329
10.36.2.156 616: . . . 329
10.36.2.157 716:displayIdle: . . . 329
10.36.2.158 816:displayIdle:displayExpiry:luhnCheck:entryMode: . . . 329
10.36.2.159 916:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout: . 329
10.36.2.160 920:optionalAmount: . . . 329
10.36.2.161 321:language:tip:cashback:surcharge: . . . 330
10.36.2.162 221prompt: . . . 330
10.36.2.163 323:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBack-Options:surchargeOptions: . . . 330
10.36.2.164 466: . . . 330
10.36.2.165 567: . . . 330
10.36.2.166 667:pinpadPrompt:timeout: . . . 331
10.36.2.167 768 . . . 332
10.36.2.168 870:language:amount:pan:mac: . . . 332
10.36.2.169 971:language:pinKey:macKey:msgLine1:msgLine2:macData: . . . 332
10.36.2.170 971Fields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK: . . 332
10.36.2.171 395 . . . 332
10.36.2.172 295Extended . . . 332
10.36.2.173 397 . . . 333
10.36.2.174 selectEncryptionMode: . . . 333
10.36.2.175 selectEncryptionMode:keyIndex: . . . 333
10.36.2.176 sendDataCommand:calcLRC: . . . 334
10.36.2.177 sendStringCommand:calcLRC: . . . 334
10.36.2.178 setAccountEntryTimeout: . . . 334
10.36.2.179 setACKTimeout: . . . 334
10.36.2.180 setAdditionalEVMFlags:ctlsMode:pdolSupport: . . . 334
10.36.2.181 setAdditionalEVMFlags:ctlsMode:pdolSupport:valueLinkCtls:amexMode: . . . 335
10.36.2.182 setC30CustomPrompts:line2:line3:line4:amt1:amt2:amt3:amt4: . . . 336
10.36.2.183 setC30DefaultPrompts . . . 337
10.36.2.184 setContinueTransactionTimeout: . . . 337
10.36.2.185 setCurrencySymbol: . . . 338
10.36.2.186 setEMVTag:value: . . . 338
10.36.2.187 setEMVTags: . . . 338
10.36.2.188 setFrameworkTimeout: . . . 339
10.36.2.189 setInitLoop: . . . 339
10.36.2.190 setIOTimer: . . . 339

10.36.2.191 setKSN20Char: . . . 339
10.36.2.192 setParameter:value: . . . 341
10.36.2.193 setParameterArray: . . . 341
10.36.2.194 setPINTimeout: . . . 341
10.36.2.195 setPromptTimeout: . . . 341
10.36.2.196 setROSParams:threshold:target:targetPercent: . . . 342
10.36.2.197 sharedController . . . 342
10.36.2.198 startServer . . . 342
10.36.2.199 statusCTLS . . . 342
10.36.2.200 stopServer . . . 343
10.36.2.201 storeTimers:pin:balance: . . . 343
10.36.2.202 testPINPad . . . 343
10.36.2.203 updateAIDRecord:AID:partialNameMatch:supportedAID:termAVN:term2ndAVN:recommendedAIDName: . . . 344
10.36.2.204 updateEMVConfigRecord:tableData: . . . 344
10.36.2.205 updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ: . . . 345
10.36.2.206 updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities: . . . 346
10.36.2.207 updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:ctlsCountry:ctlsCurrency: . . . 347
10.36.2.208 updateExtendedSchemeRecord:emvRecord:AID:maxAIDLen:appFlow:partialName:acctSelect:enableCtls:transScheme: . . . 348
10.36.2.209 updateFromUrl: . . . 349
10.36.2.210 updateFromZip: . . . 350
10.36.2.211 updateICCT:ctlsMerchantType:ctlsTermTransInfo:ctlsTermTransType:ctlsReqReceiptLimit:ctlsOptionStatus:ctlsReaderFlooorLimit: . . . 350
10.36.2.212 updatePassThru:aid:version: . . . 351
10.36.2.213 updateSchemeRecord:numberTrans:scheme:regId:CSNList:emvTableRecord:CSNFile: . . . 351
10.36.2.214 validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData: . . . 353
10.36.2.215 validateMacResponseFields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK: . . . 354
10.36.2.216 waitForResponse: . . . 355
10.36.2.217 writeToScriptFile:clear:numScripts:length:data: . . . 355
10.36.2.218 writeToSock: . . . 357
10.36.2.219 Z2:clearScreen: . . . 357
10.36.2.220 Z3:message1:message2:clearScreen: . . . 357

10.36.2.227 150:timeout:maxLen: . . . 357
10.36.2.228 250:timeout:maxLen:entryType: . . . 357
10.36.2.229 352:echo:timeout:maxLen:entryType: . . . 358
10.36.2.230 456:message1:message2: . . . 358
10.36.2.231 558:amountFormat:message1:message2: . . . 358
10.36.2.232 660: . . . 358
10.36.2.233 762:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processingMessage: . . . 358
10.36.3 Property Documentation . . . 358
10.36.3.1 accountEntryTimeout . . . 358
10.36.3.2 ackTimeout . . . 359
10.36.3.3 amountResponse . . . 359
10.36.3.4 BTconnected . . . 359
10.36.3.5 connected . . . 359
10.36.3.6 connectedSockets . . . 359
10.36.3.7 continueTransactionTimeout . . . 359
10.36.3.8 currentSocket . . . 359
10.36.3.9 delegate . . . 359
10.36.3.10 eaACC . . . 359
10.36.3.11 emaFiles . . . 359
10.36.3.12 emvTagData . . . 359
10.36.3.13 frameworkTimeout . . . 360
10.36.3.14 initialized . . . 360
10.36.3.15 ioTimer . . . 360
10.36.3.16 isGen3 . . . 360
10.36.3.17 isRunning . . . 360
10.36.3.18 isVMFRemote . . . 360
10.36.3.19 keyLoadErrorCode . . . 360
10.36.3.20 listenSocket . . . 360
10.36.3.21 pinpadConnected . . . 361
10.36.3.22 pinpadFirmwareRevision . . . 361
10.36.3.23 pinpadHardwareRevision . . . 361
10.36.3.24 pinpadManufacturer . . . 361
10.36.3.25 pinpadModelNumber . . . 361
10.36.3.26 pinpadName . . . 361
10.36.3.27 pinpadSerialNumber . . . 361
10.36.3.28 PINTimeout . . . 361
10.36.3.29 remoteMode . . . 361

10.36.3.30serverRunning . . . 361
10.36.3.31vfiCardData . . . 361
10.36.3.32vfiCipheredData . . . 362
10.36.3.33vfiDataEntry . . . 362
10.36.3.34vfiDiagnostics . . . 362
10.36.3.35vfiEMVAuthorization . . . 362
10.36.3.36vfiEMVCompletionData . . . 362
10.36.3.37vfiEMVTags . . . 362
10.36.3.38vfiEncryptionData . . . 362
10.36.3.39vsdKeyIndex . . . 362
10.37 <VFIPinpadDelegate> Protocol Reference . . . 362
10.37.1 Detailed Description . . . 363
10.37.2 Method Documentation . . . 364
10.37.2.1 btSerialData:incoming: . . . 364
10.37.2.2 pinpadConnected: . . . 364
10.37.2.3 pinpadDataReceived: . . . 364
10.37.2.4 pinpadDataSent: . . . 364
10.37.2.5 pinpadDownloadBlocks:sent: . . . 364
10.37.2.6 pinpadDownloadInfo: . . . 365
10.37.2.7 pinPadInitialized: . . . 365
10.37.2.8 pinpadLogEntry:withSeverity: . . . 365
10.37.2.9 pinpadMessage: . . . 365
10.37.2.10pinpadMSRData:expMonth:expYear:track1Data:track2Data: . . . 365
10.37.2.11pinpadMSRData:expMonth:expYear:trackData: . . . 366
10.37.2.12pinpadReconnectFinished . . . 367
10.37.2.13pinpadReconnectStarted . . . 367
10.37.2.14pinpadResultCode:result: . . . 367
10.37.2.15pinpadSerialData:incoming: . . . 367
10.37.2.16pinpadServerData:incoming: . . . 367
10.37.2.17serverConnected: . . . 368
10.38 VFISoftwareVersion Class Reference . . . 368
10.38.1 Detailed Description . . . 369
10.38.2 Method Documentation . . . 369
10.38.2.1 clear . . . 369
10.38.3 Property Documentation . . . 369
10.38.3.1 AppBuild . . . 369
10.38.3.2 AppMajor . . . 369

10.38.3.3 AppMinor . . . 369
10.38.3.4 OSBuildDay . . . 369
10.38.3.5 OSBuildMonth . . . 370
10.38.3.6 OSBuildYear . . . 370
10.38.3.7 OSID . . . 370
10.38.3.8 OSPlatform . . . 370
10.38.3.9 OSSubVersion . . . 370
10.38.3.10 OSVersion . . . 370
10.39 VFIStream Class Reference . . . 370
10.39.1 Method Documentation . . . 371
10.39.1.1 closeSession . . . 371
10.39.1.2 isBT . . . 371
10.39.1.3 isGen3 . . . 371
10.39.1.4 openSession . . . 371
10.39.1.5 readBytesAvailable . . . 371
10.39.1.6 readData: . . . 371
10.39.1.7 setBTSessionOpen: . . . 371
10.39.1.8 setupControllerForAccessory:withProtocolString: . . . 371
10.39.1.9 writeData:channel:isOpen: . . . 371
10.39.2 Member Data Documentation . . . 372
10.39.2.1 _accessory . . . 372
10.39.2.2 _protocolString . . . 372
10.39.2.3 _readData . . . 372
10.39.2.4 _session . . . 372
10.39.2.5 _sessionOpen . . . 372
10.39.2.6 _sessionOpen_Barcode . . . 372
10.39.2.7 _sessionOpen_Control . . . 372
10.39.2.8 _sessionOpen_XPI . . . 372
10.39.2.9 _sessionOpen_Zontalk . . . 372
10.39.2.10 _writeData . . . 372
10.39.3 Property Documentation . . . 372
10.39.3.1 accessory . . . 372
10.39.3.2 delegate . . . 372
10.39.3.3 protocolString . . . 372
10.39.3.4 sessionOpen . . . 372
10.39.3.5 sessionOpen_Barcode . . . 372
10.39.3.6 sessionOpen_Control . . . 372

10.39.3.7 sessionOpen_XPI . . . 372
10.39.3.8 sessionOpen_Zontalk . . . 372
10.40 <VFIStreamDelegate> Protocol Reference . . . 372
10.40.1 Method Documentation . . . 373
10.40.1.1 receivedData:protocol: . . . 373

11 File Documentation **375**
11.1 AsyncSocket.h File Reference . . . 375
11.1.1 Typedef Documentation . . . 375
11.1.1.1 AsyncSocketError . . . 375
11.1.2 Enumeration Type Documentation . . . 375
11.1.2.1 AsyncSocketError . . . 376
11.1.3 Variable Documentation . . . 376
11.1.3.1 AsyncSocketErrorDomain . . . 376
11.1.3.2 AsyncSocketException . . . 376
11.2 Barcode_host.h File Reference . . . 376
11.2.1 Macro Definition Documentation . . . 383
11.2.1.1 AIM_ID_AZTEC_RUNE . . . 383
11.2.1.2 AIM_ID_CODE11 . . . 383
11.2.1.3 AIM_ID_CODE128_ISBT_GS1 . . . 383
11.2.1.4 AIM_ID_CODE39_32 . . . 383
11.2.1.5 AIM_ID_CODE93 . . . 383
11.2.1.6 AIM_ID_CODEBAR . . . 383
11.2.1.7 AIM_ID_COMP_EC . . . 383
11.2.1.8 AIM_ID_COMP_EE . . . 383
11.2.1.9 AIM_ID_COMP_RS . . . 383
11.2.1.10 AIM_ID_DATAMATRIX . . . 383
11.2.1.11 AIM_ID_DISC2OF5 . . . 383
11.2.1.12 AIM_ID_GS1DATABAR . . . 383
11.2.1.13 AIM_ID_INTL2OF5 . . . 383
11.2.1.14 AIM_ID_MAXICODE . . . 383
11.2.1.15 AIM_ID_MSI . . . 383
11.2.1.16 AIM_ID_PDF417 . . . 383
11.2.1.17 AIM_ID_QRCODE_MICROQR . . . 383
11.2.1.18 AIM_ID_TLC39 . . . 383
11.2.1.19 AIM_ID_UPC_EAN_COUPON . . . 383
11.2.1.20 AIM_ID_X . . . 383

11.2.1.21 AUTODESCRIMINATE_SUPPLIMENTS . . . 383
11.2.1.22 CMD_AUTO_BEEP_CONFIG . . . 383
11.2.1.23 CMD_BAR_DEV_CLOSE . . . 383
11.2.1.24 CMD_BAR_DEV_OPEN . . . 384
11.2.1.25 CMD_BARCODE_DATA . . . 384
11.2.1.26 CMD_BEEP_IMMEDIATE . . . 384
11.2.1.27 CMD_DISABLE_ALL_SYMB . . . 384
11.2.1.28 CMD_EN_AIM_PATTERN . . . 384
11.2.1.29 CMD_EN_CONTINUOUS_RD . . . 384
11.2.1.30 CMD_EN_MOBILE_PH_MODE . . . 384
11.2.1.31 CMD_EN_PICKLIST_MODE . . . 384
11.2.1.32 CMD_EN_UNIQUE_CODE_REPO . . . 384
11.2.1.33 CMD_EN_XMIT_FMT . . . 384
11.2.1.34 CMD_GET_AUTO_BEEP_CONFIG . . . 384
11.2.1.35 CMD_MULTI_SCAN . . . 384
11.2.1.36 CMD_PASS_THRU . . . 384
11.2.1.37 CMD_PREFIX_VAL . . . 384
11.2.1.38 CMD_RESTORE_DEFAULTS . . . 384
11.2.1.39 CMD_SCAN_TIMEOUT . . . 384
11.2.1.40 CMD_SET_TRIG_MODE . . . 384
11.2.1.41 CMD_SINGLE_BUTTON_STATUS . . . 384
11.2.1.42 CMD_SINGLE_GET_APP_VER . . . 384
11.2.1.43 CMD_SINGLE_GET_DEVICE_ID . . . 384
11.2.1.44 CMD_SINGLE_GET_FIRM_VER . . . 384
11.2.1.45 CMD_START_SCAN . . . 384
11.2.1.46 CMD_STOP_SCAN . . . 384
11.2.1.47 CMD_SUFFIX1_VAL . . . 384
11.2.1.48 CMD_SUFFIX2_VAL . . . 384
11.2.1.49 CMD_SYMBOLOGY . . . 384
11.2.1.50 CMD_TIMEOUT_BW_DIFF_SYM . . . 385
11.2.1.51 CMD_TIMEOUT_BW_SAME_SYM . . . 385
11.2.1.52 CODE_ID_AUS_POST . . . 385
11.2.1.53 CODE_ID_AZTEC_RUNE . . . 385
11.2.1.54 CODE_ID_BOOKLAND_EAN . . . 385
11.2.1.55 CODE_ID_CHINESE2OF5 . . . 385
11.2.1.56 CODE_ID_CODABAR . . . 385
11.2.1.57 CODE_ID_CODE11 . . . 385

11.2.1.58 CODE_ID_CODE128_ISBT . . . 385
11.2.1.59 CODE_ID_CODE39_32 . . . 385
11.2.1.60 CODE_ID_CODE93 . . . 385
11.2.1.61 CODE_ID_COUPONCODE . . . 385
11.2.1.62 CODE_ID_CUECAT . . . 385
11.2.1.63 CODE_ID_DATA_MATRIX . . . 385
11.2.1.64 CODE_ID_DISC2OF5 . . . 385
11.2.1.65 CODE_ID_GS1DATABAR . . . 385
11.2.1.66 CODE_ID_GSI128 . . . 385
11.2.1.67 CODE_ID_INTL2OF5 . . . 385
11.2.1.68 CODE_ID_JAPAN_POSTAL . . . 385
11.2.1.69 CODE_ID_KOREAN3OF5 . . . 385
11.2.1.70 CODE_ID_MATRIX2OF5 . . . 385
11.2.1.71 CODE_ID_MAXICODE . . . 385
11.2.1.72 CODE_ID_MSI . . . 385
11.2.1.73 CODE_ID_NETH_KIX . . . 385
11.2.1.74 CODE_ID_PDF417_ISSNEAN . . . 385
11.2.1.75 CODE_ID_POSTBAR_CA . . . 385
11.2.1.76 CODE_ID_QRCODE_MICRO . . . 386
11.2.1.77 CODE_ID_SCANLET_WEB . . . 386
11.2.1.78 CODE_ID_TRIOPTIC39 . . . 386
11.2.1.79 CODE_ID_UCCCOMPOS . . . 386
11.2.1.80 CODE_ID_UK_POSTAL . . . 386
11.2.1.81 CODE_ID_UPC_EA . . . 386
11.2.1.82 CODE_ID_UPU_FICS . . . 386
11.2.1.83 CODE_ID_US_PLANET . . . 386
11.2.1.84 CODE_ID_US_POSTNET . . . 386
11.2.1.85 CODE_ID_USPS_4CB . . . 386
11.2.1.86 DECODE_UPC_EAN_W_SUPPLIMENTS . . . 386
11.2.1.87 DISABLE . . . 386
11.2.1.88 ENABLE . . . 386
11.2.1.89 ENABLE_378_379_SUPPLIMENTS . . . 386
11.2.1.90 ENABLE_414_419_434_439 . . . 386
11.2.1.91 ENABLE_491_SUPPLIMENTS . . . 386
11.2.1.92 ENABLE_977_SUPPLIMENT . . . 386
11.2.1.93 ENABLE_978_979_SUPPLIMENTS . . . 386
11.2.1.94 ENABLE_SMART_SUPPLIMENT . . . 386

11.2.1.95 GEN_PID_AIM_PATTERN_EN . . . 386
11.2.1.96 GEN_PID_AUTO_BEEP_ERROR . . . 386
11.2.1.97 GEN_PID_AUTO_BEEP_MODE . . . 386
11.2.1.98 GEN_PID_AUTO_BEEP_SCAN . . . 386
11.2.1.99 GEN_PID_BEEP_IMMEDIATE . . . 386
11.2.1.100 GEN_PID_CONTINUOUS_READ . . . 386
11.2.1.101 GEN_PID_MOBILE_PHONE_MODE . . . 386
11.2.1.102 GEN_PID_PASS_THRU . . . 387
11.2.1.103 GEN_PID_PICKLIST_MODE . . . 387
11.2.1.104 GEN_PID_PREFIX_KEY . . . 387
11.2.1.105 GEN_PID_PREFIX_VAL . . . 387
11.2.1.106 GEN_PID_RESTORE_DEFAULTS . . . 387
11.2.1.107 GEN_PID_SCAN_DATA_XMIT_FMT . . . 387
11.2.1.108 GEN_PID_SCAN_TIMEOUT . . . 387
11.2.1.109 GEN_PID_SET_TRIG_MODE . . . 387
11.2.1.110 GEN_PID_SUFFIX1_KEY . . . 387
11.2.1.111 GEN_PID_SUFFIX1_VAL . . . 387
11.2.1.112 GEN_PID_SUFFIX2_KEY . . . 387
11.2.1.113 GEN_PID_SUFFIX2_VAL . . . 387
11.2.1.114 GEN_PID_TIMEOUT_BW_DIFF_SYM . . . 387
11.2.1.115 GEN_PID_TIMEOUT_BW_SAME_SYM . . . 387
11.2.1.116 GEN_PID_UNIQUE_CODE_REPORT . . . 387
11.2.1.117 HOST_BUF_SIZE . . . 387
11.2.1.118 IGNORE_UPC_EAN_W_SUPPLIMENTS . . . 387
11.2.1.119 RESP_ACK . . . 387
11.2.1.120 RESP_BARCODE_DATA . . . 387
11.2.1.121 RESP_BUTTON_STATUS . . . 387
11.2.1.122 RESP_NAK . . . 387
11.2.1.123 SMART_SUPPLIMENT_USER_PROG1 . . . 387
11.2.1.124 SMART_SUPPLIMENT_USER_PROG1_2 . . . 387
11.2.1.125 SUPPLIMENT_USER_PROG1 . . . 387
11.2.1.126 SUPPLIMENT_USER_PROG1_2 . . . 387
11.2.1.127 SYM_ID_BOOKLAND . . . 387
11.2.1.128 SYM_ID_CCA_EAN128 . . . 388
11.2.1.129 SYM_ID_CCA_EAN13 . . . 388
11.2.1.130 SYM_ID_CCA_EAN8 . . . 388
11.2.1.131 SYM_ID_CCA_RSS14 . . . 388

11.2.1.132SYM_ID_CCA_RSSEXPAND 388
11.2.1.133SYM_ID_CCA_RSSLIMIT 388
11.2.1.134SYM_ID_CCA_UPCA 388
11.2.1.135SYM_ID_CCA_UPCE 388
11.2.1.136SYM_ID_CCB_EAN128 388
11.2.1.137SYM_ID_CCB_EAN13 388
11.2.1.138SYM_ID_CCB_EAN8 388
11.2.1.139SYM_ID_CCB_RSS14 388
11.2.1.140SYM_ID_CCB_RSSEXPAND 388
11.2.1.141SYM_ID_CCB_RSSLIMIT 388
11.2.1.142SYM_ID_CCB_UPCA 388
11.2.1.143SYM_ID_CCB_UPCE 388
11.2.1.144SYM_ID_CCC_EAN128 388
11.2.1.145SYM_ID_CODABAR 388
11.2.1.146SYM_ID_CODE11 388
11.2.1.147SYM_ID_CODE128 388
11.2.1.148SYM_ID_CODE32 388
11.2.1.149SYM_ID_CODE39 388
11.2.1.150SYM_ID_CODE39FULL 388
11.2.1.151SYM_ID_CODE93 388
11.2.1.152SYM_ID_COUPONCODE 388
11.2.1.153SYM_ID_CUECAT 388
11.2.1.154SYM_ID_D25 389
11.2.1.155SYM_ID_DATAMATRIX 389
11.2.1.156SYM_ID_EAN128 389
11.2.1.157SYM_ID_EAN13 389
11.2.1.158SYM_ID_EAN13_2 389
11.2.1.159SYM_ID_EAN13_5 389
11.2.1.160SYM_ID_EAN8 389
11.2.1.161SYM_ID_EAN8_2 389
11.2.1.162SYM_ID_EAN8_5 389
11.2.1.163SYM_ID_IATA 389
11.2.1.164SYM_ID_ISBT128 389
11.2.1.165SYM_ID_ISBT128CONC 389
11.2.1.166SYM_ID_ITF 389
11.2.1.167SYM_ID_KOR3OF5 389
11.2.1.168SYM_ID_MACROPDF 389

11.2.1.169 SYM_ID_MACROPDF417 . . . 389
11.2.1.170 SYM_ID_MAXICODE . . . 389
11.2.1.171 SYM_ID_MICROPDF . . . 389
11.2.1.172 SYM_ID_MSI . . . 389
11.2.1.173 SYM_ID_PDF417 . . . 389
11.2.1.174 SYM_ID_PLANETUS . . . 389
11.2.1.175 SYM_ID_POSTALAUST . . . 389
11.2.1.176 SYM_ID_POSTALDUTCH . . . 389
11.2.1.177 SYM_ID_POSTALJAPAN . . . 389
11.2.1.178 SYM_ID_POSTALUK . . . 389
11.2.1.179 SYM_ID_POSTBARCA . . . 389
11.2.1.180 SYM_ID_POSTNETUS . . . 390
11.2.1.181 SYM_ID_QRCODE . . . 390
11.2.1.182 SYM_ID_RSS14 . . . 390
11.2.1.183 SYM_ID_RSSEXPAND . . . 390
11.2.1.184 SYM_ID_RSSLIMIT . . . 390
11.2.1.185 SYM_ID_SCANLETWEB . . . 390
11.2.1.186 SYM_ID_TLC39 . . . 390
11.2.1.187 SYM_ID_TRIOPTIC . . . 390
11.2.1.188 SYM_ID_UPCA . . . 390
11.2.1.189 SYM_ID_UPCA_2 . . . 390
11.2.1.190 SYM_ID_UPCA_5 . . . 390
11.2.1.191 SYM_ID_UPCE . . . 390
11.2.1.192 SYM_ID_UPCE1 . . . 390
11.2.1.193 SYM_ID_UPCE1_2 . . . 390
11.2.1.194 SYM_ID_UPCE1_5 . . . 390
11.2.1.195 SYM_ID_UPCE_2 . . . 390
11.2.1.196 SYM_ID_UPCE_5 . . . 390
11.2.1.197 SYM_PID_CHECK_ISBT_TABLE . . . 390
11.2.1.198 SYM_PID_CODE11_CHK_DIGIT . . . 390
11.2.1.199 SYM_PID_CODE32_PREFIX . . . 390
11.2.1.200 SYM_PID_CODE39_BUFFERING . . . 390
11.2.1.201 SYM_PID_CODE39_CHK_DIGIT . . . 390
11.2.1.202 SYM_PID_CODE39_FULL_ASCII . . . 390
11.2.1.203 SYM_PID_CONV_CODE39_2_CODE32 . . . 390
11.2.1.204 SYM_PID_CONV_I2OF5_EAN13 . . . 390
11.2.1.205 SYM_PID_CONVERT_UPCE1_2_UPCA . . . 390

11.2.1.206 SYM_PID_CONVERT_UPCE_2_UPCA . . . 391
11.2.1.207 SYM_PID_COUPON_REPORT . . . 391
11.2.1.208 SYM_PID_EAN8_JAN8_EXTEND . . . 391
11.2.1.209 SYM_PID_EN_AUST_POST_FMT . . . 391
11.2.1.210 SYM_PID_EN_AUSTRALIA_POST . . . 391
11.2.1.211 SYM_PID_EN_AZTEC . . . 391
11.2.1.212 SYM_PID_EN_AZTEC_INVERSE . . . 391
11.2.1.213 SYM_PID_EN_BOOKLAND_EAN . . . 391
11.2.1.214 SYM_PID_EN_CHINESE2OF5 . . . 391
11.2.1.215 SYM_PID_EN_CLSI . . . 391
11.2.1.216 SYM_PID_EN_CODABAR . . . 391
11.2.1.217 SYM_PID_EN_CODE11 . . . 391
11.2.1.218 SYM_PID_EN_CODE128 . . . 391
11.2.1.219 SYM_PID_EN_CODE39 . . . 391
11.2.1.220 SYM_PID_EN_CODE93 . . . 391
11.2.1.221 SYM_PID_EN_COMP_CC_A_B . . . 391
11.2.1.222 SYM_PID_EN_COMP_CC_C . . . 391
11.2.1.223 SYM_PID_EN_COMP_TLC_39 . . . 391
11.2.1.224 SYM_PID_EN_CONV_UPC_EAN . . . 391
11.2.1.225 SYM_PID_EN_DATA_MATRIX . . . 391
11.2.1.226 SYM_PID_EN_DATA_MATRIX_INVERSE . . . 391
11.2.1.227 SYM_PID_EN_DISC2OF5 . . . 391
11.2.1.228 SYM_PID_EN_EAN13_JAN13 . . . 391
11.2.1.229 SYM_PID_EN_EAN8_JAN8 . . . 391
11.2.1.230 SYM_PID_EN_GS1128 . . . 391
11.2.1.231 SYM_PID_EN_GS1_128_EMULATION . . . 391
11.2.1.232 SYM_PID_EN_GS1_DATABAR . . . 392
11.2.1.233 SYM_PID_EN_GS1_EXPANDED . . . 392
11.2.1.234 SYM_PID_EN_GS1_LIMITED . . . 392
11.2.1.235 SYM_PID_EN_INTER2OF5 . . . 392
11.2.1.236 SYM_PID_EN_INVERSE1D . . . 392
11.2.1.237 SYM_PID_EN_ISBN_FORMAT . . . 392
11.2.1.238 SYM_PID_EN_ISBT . . . 392
11.2.1.239 SYM_PID_EN_JAPAN_POSTAL . . . 392
11.2.1.240 SYM_PID_EN_KOREAN3OF5 . . . 392
11.2.1.241 SYM_PID_EN_MATRIX2OF5 . . . 392
11.2.1.242 SYM_PID_EN_MAXICODE . . . 392

11.2.1.243SYM_PID_EN_MICRO_PDF417 . . . 392
11.2.1.244SYM_PID_EN_MICRO_QR . . . 392
11.2.1.245SYM_PID_EN_MIRROR_IMAGES . . . 392
11.2.1.246SYM_PID_EN_MSI . . . 392
11.2.1.247SYM_PID_EN_NETHERLANDS_KIX . . . 392
11.2.1.248SYM_PID_EN_NOTIS . . . 392
11.2.1.249SYM_PID_EN_PDF417 . . . 392
11.2.1.250SYM_PID_EN_QR_CODE . . . 392
11.2.1.251SYM_PID_EN_QR_INVERSE . . . 392
11.2.1.252SYM_PID_EN_TRIOPTIC_CODE39 . . . 392
11.2.1.253SYM_PID_EN_UK_POSTAL . . . 392
11.2.1.254SYM_PID_EN_UPCA . . . 392
11.2.1.255SYM_PID_EN_UPCE . . . 392
11.2.1.256SYM_PID_EN_UPCE1 . . . 392
11.2.1.257SYM_PID_EN_UPU_FICS . . . 392
11.2.1.258SYM_PID_EN_US_PLANET . . . 393
11.2.1.259SYM_PID_EN_US_POSTNET . . . 393
11.2.1.260SYM_PID_EN_USPS_4CB . . . 393
11.2.1.261SYM_PID_I2OF5_CHECK_DIGIT . . . 393
11.2.1.262SYM_PID_ISBT_CONCATE . . . 393
11.2.1.263SYM_PID_ISBT_CONCATE_REDUN . . . 393
11.2.1.264SYM_PID_ISSN_EAN . . . 393
11.2.1.265SYM_PID_LTD_SECURITY . . . 393
11.2.1.266SYM_PID_M2OF5_CHK_DIGIT . . . 393
11.2.1.267SYM_PID_MSI_CHK_DIGIT . . . 393
11.2.1.268SYM_PID_MSI_CHK_DIGIT_ALGOR . . . 393
11.2.1.269SYM_PID_SETLEN_1DISCRETE_C11 . . . 393
11.2.1.270SYM_PID_SETLEN_1DISCRETE_C128 . . . 393
11.2.1.271SYM_PID_SETLEN_1DISCRETE_C39 . . . 393
11.2.1.272SYM_PID_SETLEN_1DISCRETE_C93 . . . 393
11.2.1.273SYM_PID_SETLEN_1DISCRETE_CBAR . . . 393
11.2.1.274SYM_PID_SETLEN_1DISCRETE_D2OF5 . . . 393
11.2.1.275SYM_PID_SETLEN_1DISCRETE_I2OF5 . . . 393
11.2.1.276SYM_PID_SETLEN_1DISCRETE_M2OF5 . . . 393
11.2.1.277SYM_PID_SETLEN_1DISCRETE_MSI . . . 393
11.2.1.278SYM_PID_SETLEN_2DISCRETE_C11 . . . 393
11.2.1.279SYM_PID_SETLEN_2DISCRETE_C128 . . . 393

11.2.1.280 SYM_PID_SETLEN_2DISCRETE_C39 . . . 393
11.2.1.281 SYM_PID_SETLEN_2DISCRETE_C93 . . . 393
11.2.1.282 SYM_PID_SETLEN_2DISCRETE_CBAR . . . 393
11.2.1.283 SYM_PID_SETLEN_2DISCRETE_D2OF5 . . . 393
11.2.1.284 SYM_PID_SETLEN_2DISCRETE_I2OF5 . . . 394
11.2.1.285 SYM_PID_SETLEN_2DISCRETE_M2OF5 . . . 394
11.2.1.286 SYM_PID_SETLEN_2DISCRETE_MSI . . . 394
11.2.1.287 SYM_PID_SETLEN_ANY_C11 . . . 394
11.2.1.288 SYM_PID_SETLEN_ANY_C128 . . . 394
11.2.1.289 SYM_PID_SETLEN_ANY_C39 . . . 394
11.2.1.290 SYM_PID_SETLEN_ANY_C93 . . . 394
11.2.1.291 SYM_PID_SETLEN_ANY_CBAR . . . 394
11.2.1.292 SYM_PID_SETLEN_ANY_D2OF5 . . . 394
11.2.1.293 SYM_PID_SETLEN_ANY_I2OF5 . . . 394
11.2.1.294 SYM_PID_SETLEN_ANY_M2OF5 . . . 394
11.2.1.295 SYM_PID_SETLEN_ANY_MSI . . . 394
11.2.1.296 SYM_PID_SETLEN_RANGE_C11 . . . 394
11.2.1.297 SYM_PID_SETLEN_RANGE_C128 . . . 394
11.2.1.298 SYM_PID_SETLEN_RANGE_C39 . . . 394
11.2.1.299 SYM_PID_SETLEN_RANGE_C93 . . . 394
11.2.1.300 SYM_PID_SETLEN_RANGE_CBAR . . . 394
11.2.1.301 SYM_PID_SETLEN_RANGE_D2OF5 . . . 394
11.2.1.302 SYM_PID_SETLEN_RANGE_I2OF5 . . . 394
11.2.1.303 SYM_PID_SETLEN_RANGE_M2OF5 . . . 394
11.2.1.304 SYM_PID_SETLEN_RANGE_MSI . . . 394
11.2.1.305 SYM_PID_START_STOP_CASE . . . 394
11.2.1.306 SYM_PID_SUPPLIMENTS_UPC_EAN . . . 394
11.2.1.307 SYM_PID_UCC_COUPON_EXTEND . . . 394
11.2.1.308 SYM_PID_UPC_COMP_MODE . . . 394
11.2.1.309 SYM_PID_UPC_EAN_JAN_SUPP_REDUN . . . 394
11.2.1.310 SYM_PID_USER_PROG_SUPP1 . . . 395
11.2.1.311 SYM_PID_USER_PROG_SUPP2 . . . 395
11.2.1.312 SYM_PID_XMIT_CODE11_CHK_DIGIT . . . 395
11.2.1.313 SYM_PID_XMIT_CODE39_CHK_DIGIT . . . 395
11.2.1.314 SYM_PID_XMIT_I2OF5_CHECK_DIGIT . . . 395
11.2.1.315 SYM_PID_XMIT_M2OF5_CHK_DIGIT . . . 395
11.2.1.316 SYM_PID_XMIT_MSI_CHK_DIGIT . . . 395

11.2.1.317 SYM_PID_XMIT_UK_POST_CHK_DIGIT . . . 395
11.2.1.318 SYM_PID_XMIT_UPCA_CHECK_DIGIT . . . 395
11.2.1.319 SYM_PID_XMIT_UPCA_PREAMBLE . . . 395
11.2.1.320 SYM_PID_XMIT_UPCE1_CHECK_DIGIT . . . 395
11.2.1.321 SYM_PID_XMIT_UPCE1_PREAMBLE . . . 395
11.2.1.322 SYM_PID_XMIT_UPCE_CHECK_DIGIT . . . 395
11.2.1.323 SYM_PID_XMIT_UPCE_PREAMBLE . . . 395
11.2.1.324 SYM_PID_XMIT_US_POST_CHK_DIGIT . . . 395
11.2.2 Variable Documentation . . . 395
11.2.2.1 symb_AIM . . . 395
11.3 decryptedCredentials.h File Reference . . . 395
11.4 GlobalMarkup.d File Reference . . . 395
11.5 ISCP_HIGH.h File Reference . . . 396
11.6 swipeData.h File Reference . . . 396
11.6.1 Enumeration Type Documentation . . . 396
11.6.1.1 CardType . . . 396
11.7 VFIAccessoryMgr.h File Reference . . . 397
11.7.1 Variable Documentation . . . 399
11.7.1.1 CloseSession_Gen20_Barcode . . . 399
11.7.1.2 CloseSession_Gen20_Control . . . 399
11.7.1.3 CloseSession_Gen20_XPI . . . 399
11.7.1.4 CloseSession_Gen20_Zontalk . . . 399
11.7.1.5 CloseSession_Gen25 . . . 399
11.7.1.6 CloseSession_Gen30_Barcode . . . 399
11.7.1.7 CloseSession_Gen30_Control . . . 399
11.7.1.8 CloseSession_Gen30_XPI . . . 399
11.7.1.9 CloseSession_Gen30_Zontalk . . . 399
11.7.1.10 CloseSession_Printer . . . 399
11.7.1.11 CloseSession_Sled . . . 399
11.7.1.12 Connected_Gen20_Barcode . . . 399
11.7.1.13 Connected_Gen20_Control . . . 399
11.7.1.14 Connected_Gen20_XPI . . . 399
11.7.1.15 Connected_Gen20_Zontalk . . . 399
11.7.1.16 Connected_Gen25 . . . 399
11.7.1.17 Connected_Gen30_Barcode . . . 399
11.7.1.18 Connected_Gen30_Control . . . 399
11.7.1.19 Connected_Gen30_XPI . . . 399

11.7.1.20 Connected_Gen30_Zontalk . . . 399
11.7.1.21 Connected_Printer . . . 399
11.7.1.22 Connected_Sled . . . 399
11.7.1.23 DataReceivedNotification_Barcode . . . 399
11.7.1.24 DataReceivedNotification_Barcode_BT . . . 399
11.7.1.25 DataReceivedNotification_Control . . . 399
11.7.1.26 DataReceivedNotification_Control_BT . . . 400
11.7.1.27 DataReceivedNotification_Printer . . . 400
11.7.1.28 DataReceivedNotification_Sled . . . 400
11.7.1.29 DataReceivedNotification_XPI . . . 400
11.7.1.30 DataReceivedNotification_XPI_BT . . . 400
11.7.1.31 DataReceivedNotification_Zontalk . . . 400
11.7.1.32 DataReceivedNotification_Zontalk_BT . . . 400
11.7.1.33 Disconnected_Gen20_Barcode . . . 400
11.7.1.34 Disconnected_Gen20_Control . . . 400
11.7.1.35 Disconnected_Gen20_XPI . . . 400
11.7.1.36 Disconnected_Gen20_Zontalk . . . 400
11.7.1.37 Disconnected_Gen25 . . . 400
11.7.1.38 Disconnected_Gen30_Barcode . . . 400
11.7.1.39 Disconnected_Gen30_Control . . . 400
11.7.1.40 Disconnected_Gen30_XPI . . . 400
11.7.1.41 Disconnected_Gen30_Zontalk . . . 400
11.7.1.42 Disconnected_Printer . . . 400
11.7.1.43 Disconnected_Sled . . . 400
11.7.1.44 OpenSession_Gen20_Barcode . . . 400
11.7.1.45 OpenSession_Gen20_Control . . . 400
11.7.1.46 OpenSession_Gen20_XPI . . . 400
11.7.1.47 OpenSession_Gen20_Zontalk . . . 400
11.7.1.48 OpenSession_Gen25 . . . 400
11.7.1.49 OpenSession_Gen25_Barcode . . . 400
11.7.1.50 OpenSession_Gen25_Control . . . 400
11.7.1.51 OpenSession_Gen25_XPI . . . 400
11.7.1.52 OpenSession_Gen25_Zontalk . . . 401
11.7.1.53 OpenSession_Gen30_Barcode . . . 401
11.7.1.54 OpenSession_Gen30_Control . . . 401
11.7.1.55 OpenSession_Gen30_XPI . . . 401
11.7.1.56 OpenSession_Gen30_Zontalk . . . 401

11.7.1.57 OpenSession_Printer . . . 401
11.7.1.58 OpenSession_Sled . . . 401
11.7.1.59 Protocol_Gen20_Barcode . . . 401
11.7.1.60 Protocol_Gen20_Control . . . 401
11.7.1.61 Protocol_Gen20_XPI . . . 401
11.7.1.62 Protocol_Gen20_Zontalk . . . 401
11.7.1.63 Protocol_Gen25 . . . 401
11.7.1.64 Protocol_Gen30_Barcode . . . 401
11.7.1.65 Protocol_Gen30_Control . . . 401
11.7.1.66 Protocol_Gen30_XPI . . . 401
11.7.1.67 Protocol_Gen30_Zontalk . . . 401
11.7.1.68 Protocol_Printer . . . 401
11.7.1.69 Protocol_Sled . . . 401
11.7.1.70 UpdateEA_Barcode . . . 401
11.7.1.71 UpdateEA_Control . . . 401
11.7.1.72 UpdateEA_Printer . . . 401
11.7.1.73 UpdateEA_Sled . . . 401
11.7.1.74 UpdateEA_XPI . . . 401
11.7.1.75 UpdateEA_Zontalk . . . 401
11.8 VFIBarcode.h File Reference . . . 402
11.8.1 Enumeration Type Documentation . . . 403
11.8.1.1 barcodeTypes . . . 403
11.8.1.2 commResult . . . 403
11.8.1.3 scanningModes . . . 404
11.8.1.4 triggerResponse . . . 404
11.8.1.5 TYPE . . . 404
11.9 VFIBTBridge.h File Reference . . . 405
11.10 VFICardData.h File Reference . . . 405
11.11 VFICipheredData.h File Reference . . . 405
11.12 VFIControl.h File Reference . . . 406
11.13 VFIDataEntry.h File Reference . . . 406
11.14 VFIDecryptedCredentials.h File Reference . . . 406
11.15 VFIDiagnostics.h File Reference . . . 406
11.16 VFIEMAFiles.h File Reference . . . 407
11.17 VFIEMVAuthorization.h File Reference . . . 407
11.18 VFIEMVCompletionData.h File Reference . . . 407
11.19 VFIEMVConfiguration.h File Reference . . . 407

11.20 VFIEMVResponse.h File Reference . . . 408
11.21 VFIEMVTags.h File Reference . . . 408
11.22 VFIEncryptedBarcode.h File Reference . . . 408
11.23 VFIEncryptionData.h File Reference . . . 408
11.24 VFIFinancialData.h File Reference . . . 409
11.25 VFIGen3DeviceInfo.h File Reference . . . 409
11.26 VFIKeypadVersion.h File Reference . . . 409
11.27 VFIPinpad.h File Reference . . . 409
11.27.1 Enumeration Type Documentation . . . 410
11.27.1.1 EncryptionMode . . . 411
11.27.1.2 SEVERITY_LOG_LEVELS . . . 411
11.27.1.3 VFIErrorCode . . . 411
11.28 VFISoftwareVersion.h File Reference . . . 411
11.29 VFIStream.h File Reference . . . 412
11.30 VFISwipeData.h File Reference . . . 412
11.30.1 Enumeration Type Documentation . . . 412
11.30.1.1 CardType . . . 412

Index **414**

Chapter 1

VeriFone Framework Reference Guide

The VeriFone Mobile Framework (VMF) and VeriFone Mobile Communication Framework (VCMF) are Apple iOS Frameworks that will be provided by VeriFone as the main interface between iOS applications and VeriFone hardware and payment processing solutions.

The purpose of this document is to describe the requirements of the frameworks as well as the interface definitions and requirements needed for any iPhone applications wishing to deploy with the payment application.

- VMF API Reference Guide
- VCMF API Reference Guide
- e315 Barcode Reference
- e210/e255/e315 Configuration Parameters

Chapter 2

e210/e255/e315 Configuration Parameters

The following table are valid configuration parameters for Vx Solutions and VX Evolution Devices, including the e210, e255, and e315:

Parameter	Description	Values
ENTRYMODE	Determines whether the device will allow manual account number/ expiration date entry, swipe data entry, or both.	0 - (Default) Both manual entry and card swipe supported, 1 - Manual entry only is supported, 2 - Swipe only is supported
PROMPTO	Timeout value for Prompts, in seconds.	Numeric. 0-99. Default 15.
DIGITO	Timeout value for Inter-digit PIN Entry.	Numeric. 0-99
BALTO	Timeout value for Display Balance.	Numeric. 0-99
CBMODE	Turns on/off the cash back entry prompt for the S21 command. A value of 1 displays cash back entry prompt. A value of 0 displays amounts set in CBAMT1, CBAMT2, CBAMT3, and CBAMT4.	Flag. 0/1
CHQ	Sets the checking account value.	Alphanumeric. 1 / A
SAV	Sets the saving account value.	Alphanumeric. 2 / B
EMVDS	Determines if C34 command displays approved/declined if there is no optional show status parameter sent in the C34 command. A value of 1 means don't display, a value of 0 means display.	Flag. 0/1

CBAMT1	Cash back amount 1 for S21 command.	Numeric. 0-9999
CBAMT2	Cash back amount 2 for S21 command.	Numeric. 0-9999
CBAMT3	Cash back amount 3 for S21 command.	Numeric. 0-9999
CBAMT4	Cash back amount 4 for S21 command.	Numeric. 0-9999
EMVLANG	Sets the EMV default language.	Flag. 0/1
DEFLANG	Sets the terminal default language.	Alpha. E/F
SNREMOVE	Sets the number of characters to be removed from the beginning of the 9 digit device serial number.	Numeric. 0-9
SNPREFIX	Sets the prefix to be added in front of device serial number.	Alphanumeric. 3
BACKLITE	Turns on/off Back light option.	Flag. 0/1
VISATTQ	Sets the VISA TTQ.	Hexadecimal. 8
BAUDRATE	Sets the default Baud Rate.	300/1200/2400/9600/115200
FORMAT	Sets the parity setting.	8N1/7E1
COMMPORT	Sets the default Comm Port.	2 - RS232, 6 - USB using the bas, U - USB, I - Ethernet, M - MxPipe. NOTE: CORR + 8 is disabled, if M is the current XPI Comm port setting.
URLHOST	Sets the IP Address of the host.	String: IPv4. Ex: 123.123.123.123
PORT	Host (ECR) Port Number used in IP connection. Leave this field empty to disable checking against the Host Port Number.	String: 5. Valid Range: 0-65535
WELCMSG	Sets the IDLE message display.	16 Alphanumeric
COUNTRY	Sets the country.	Alphanumeric. CANADA or US
PELAGE	If set to 1, and the country code is not available, the application will check if another AID on the card is the Interac AID. If so, the card is determined to be Canadian.	0 - Do not check, 1 - Check
TERMPW	VMAC entry password.	Alphanumeric. 6
LANGMODE	Sets the language mode to Unilingual or Bilingual. When set to Unilingual, prompts that can be displayed in two languages will only be displayed in one language.	Numeric. 1 - Unilingual, 0 or missing - Bilingual
APPCONFRM	Sets an additional application confirmation prompt for EMV transactions when application confirmation is required by the card.	0 - Do not prompt, 1 - Prompt if card requires prompting.

XPILANGCNT	Number of languages supported.	Numeric. 2-5. Default is 2.
XPILANG0	First default language.	Alpha. Default is ENGL.
XPILANG1	Second default language.	Alpha. Default is FREN.
XPILANG2	ISO Code for third default language.	Alpha.
XPILANG3	ISO Code for fourth default language.	Alpha.
XPILANG4	ISO Code for fifth default language.	Alpha.
XPIDUKPTIDX	Identifies the DUKPT engine used by XPI.	Numeric: 0, 1, or 2 (via packet 19). Default is 0.
XPITO	XPI Sleep Timeout for battery-operated terminals, like PAYware Mobile and Vx 670. Expressed in seconds.	If not present: Default value = 180 seconds (3 minutes)
ICCERR	Sets number of times an inserted card failure occurs before returning the error back to the host device.	Type: 1N, Values: 0-9, Default: 0
LINE1	Line 1 Display	Alpha.
LINE2	Line 2 Display	Alpha.
LINE3	Line 3 Display	Alpha.
LINE4	Line 4 Display	Alpha.
OPTFLAG	Optional flag for additional EMV information (D25)	First Byte: 0 - Visa Debit support disabled, 1 - Visa Debit support enabled. Second Byte: 0 - Contactless EMV, 1 - Contactless MSD only, 2 - Disable Contactless. Third Byte: 0 - PDOL Support disabled, 1 - PDOL Support enabled. Fourth Byte: 0 - Valuelink disabled, 1 - Valuelink enabled. Fifth Byte: 0 - ISIS NFC disabled, 1 - ISIS NFC enabled. Sixth Byte: AMEX Contactless Mode, 1 - CTLS MSD Mode, 2 - CTLS EMV Mode. Seventh - Sixteenth Byte: Reserved for future use

XPIDEFSEC	Sets the default encryption mode during application start-up. NOTE: If E10 command is sent to change the encryption mode, this parameter will be updated as well.	Type: String, NOE = No Encryption, VSP = VeriShield Protect, PKI = Vx Public Key Infrastructure. Default = VSP
XPIALIGN	Sets the alignment of PIN during entry by the user. When XPIALIGN is set to Center, PIN alignment is center when running C32 on all terminals, and S23, S67, S68, S70, Z60, and Z62 on the e210/e255/e315. However, PIN alignment is left, even if XPIALIGN is set to Center, when running S23, S67, S68, S70, Z60, and Z62 on other terminals.	0 = Left, 1 = Center. Default = Left
MACMODE	Allow a processor to specify if MACing will occur on the encrypted PAN or the unencrypted PAN whenever VSP is enabled on a device. This value is set by the processor / acquirer based on their business requirements for account number MACing when the application is downloaded. The parameter turns the MAC decryption on / off. The default value is off, meaning MACing is done against the encrypted PAN. NOTE: The MACMODE parameter will affect the S66 and S70 commands.	Type: 1N, 0 - MAC Decryption is turned off. MACing occurs against the encrypted PAN. 1 - Decryption is turned on. MACing occurs against the unencrypted PAN. Default: 0
XPIFORCECTLS	This setting is used to determine the behavior at startup as it relates to contactless initialization. The variable can be set to always initialize the contactless reader at start-up (1) or only initialize on first start-up (0). NOTE: Contactless initialization can still be conducted during runtime via the D18 or D25 commands regardless of the setting for this variable.	Type: 1N, 0 - Initialize only at first start-up, 1 - Initialize at every start-up. Default: 1 - Always Initialize

XPISVCCODECLR	This variable is used to determine how the service code portion of the card data will be treated when PKI encryption is used. Normally, when PKI encryption is enabled, all of the card data is encrypted. Setting this variable to 1 allows the service code to be returned in the clear.	Type: 1N, 0 - Service Code will be Randomize (Default), 1 - Service Code will be in Clear (Not Encrypted). Default: 0
XPIMINPAN	This variable allows the configuration of the minimum PAN length to be set between 6 and 13 digits.	Type: 2N, Values: 6 - 13, Default: 13
XPIMAXPAN	This variable allows the configuration of the maximum PAN length to be set between 13 and 25 digits.	Type: 2N, Values: 13 - 25. Default: 19
XPIKSNPAD	When enabled, prepends the F character to the KSN to a total length of twenty characters.	Flag. 0/1, 0 = Off, 1 = On. Default: 0
XPIENCRPT	This variable was added to allow previously country-dependent commands to be used as generic commands. Certain commands (Z60 / Z62 for example) are now processed based on the encryption type rather than the COUNTRY setting. For additional information, see Terminal User Interface, Debit Transaction for commands affected.	Type: String, Values: DUKPT, MS1DES, MS3DES, INTERAC. Default: DUKPT
XPICURR	Used to set the currency symbol.	Type: 3AN, Values: $, , , . Default: $
XPIPANCHK	Validates that PAN on Tag5A is the same as PAN on Tag57.	Type: 1N, Values: 1, 0. Default: 0
XPIFONTSIZE	Supports two fonts when running on the VX 805 and VX 520 terminals	Type: STRING, Values: TRANS to use ACTION_FONT_NAME; MENU to use for MENU_FONT_NAME
XPIACCNTMASK	The Manual Entry of the account number using the S16 command will allow for the account number to be masked during entry.	Type: 1N, Values: 1, 0. Default: 0
XPILATENT	Latency of the XPI application can be set. This should be set to 1, so that XPI will not be displayed in the VMAC menu.	Type: 1/N, Values: 1, 0, Default: 0
XPILATENTCODE	Sets the key code to activate the application from the menu.	Type: 1N, Values: 0-9, Default: 0

Chapter 3

e315 Barcode Reference

The following methods are for the exclusive use of e315 Barcode Scanner:

- mDeviceOpen (VFIBarcode)
- mDeviceClose (VFIBarcode)
- mStartScan (VFIBarcode)
- mStopScan (VFIBarcode)
- mTriggerMode: (VFIBarcode)
- mBeepImmediate: (VFIBarcode)
- mAppVersion (VFIBarcode)
- mFirmwareVersion (VFIBarcode);
- mAutoBeepConfig: (VFIBarcode);
- mAutoBeepConfig:scanBeep:errorBeep: (VFIBarcode)
- mButtonStatus (VFIBarcode)
- mRestoreDefaults (VFIBarcode)
- mPicklistMode: (VFIBarcode)
- mScanTimeout: (VFIBarcode)
- mDifferentSymbolDecodesTimeout: (VFIBarcode)
- mContinuousMode: (VFIBarcode)
- mUniqueCodeReport: (VFIBarcode)
- mElectronicDisplayRead: (VFIBarcode)
- mGetAutoBeepMode (VFIBarcode)

- mGetAutoBeepScan (VFIBarcode)
- mGetAutoBeepError (VFIBarcode)
- mScanDataPrefix:value: (VFIBarcode)
- mScanDataSuffix1:value: (VFIBarcode)
- mScanDataSuffix2:value: (VFIBarcode)
- mScanDataTransmitFormat: (VFIBarcode)
- mAimPattern: (VFIBarcode)
- mSymbology:value: (VFIBarcode)
- mSymbology:value: (VFIBarcode)
- mDisableAllSymbologies (VFIBarcode)

3.1 Reference

- Beeper Tone Definitions
- Default Parameter Values
- Barcode_host.h reference file

3.2 Beeper Tone Definitions

The beeper device supports 96 distinct tones designed to approximate eight octaves of the equal tempered musical scale of standard international pitch, with treble A having a frequency of 440 Hz. Actual frequencies generated are shown in Table 1 along with the corresponding musical notes and variations. Table 1 reflects a system frequency of 200 MHz, the maximum duration in 10,000 ms or 10 seconds.

The column labels indicate the following characteristics for each of the 96 notes:

- Note standard do-re-mi designation for the musical note.
- N# the note number, used as the frequency parameter in T_BEEP_DEF and M_BEEP_DEF structure.
- Nominal frequency in Hertz for the standard musical note.

Note	N#	Nominal	Note	N#	Nominal
A	0	55.00	A	48	880
A#	1	58.27	A#	49	932
B	2	61.74	B	50	988
C	3	65.41	C	51	1047
C#	4	69.30	C#	52	1109
D	5	73.42	D	53	1175
D#	6	77.78	D#	54	1245

E	7	82.41	E	55	1319
F	8	87.31	F	56	1397
F#	9	92.50	F#	57	1480
G	10	98.00	G	58	1568
G#	11	103.83	G#	59	1661
A	12	110.00	A	60	1760
A#	13	116.54	A#	61	1865
B	14	123.47	B	62	1976
C	15	130.81	C	63	2093
C#	16	138.59	C#	64	2217
D	17	146.83	D	65	2349
D#	18	155.56	D#	66	2489
E	19	164.81	E	67	2637
F	20	174.61	F	68	2794
F#	21	185.00	F#	69	2960
G	22	196.00	G	70	3136
G#	23	207.65	G#	71	3322
A	24	220.00	A	72	3520
A#	25	233.08	A#	73	3729
B	26	246.94	B	74	3951
C	27	261.63	C	75	4186
C#	28	277.18	C#	76	4435
D	29	293.66	D	77	4699
D#	30	311.13	D#	78	4978
E	31	329.63	E	79	5274
F	32	349.23	F	80	5588
F#	33	369.99	F#	81	5920
G	34	392.00	G	82	6272
G#	35	415.30	G#	83	6645
A	36	440.00	A	84	7040
A#	37	466.16	A#	85	7459
B	38	493.88	B	86	7902
C	39	523.25	C	87	8372
C#	40	554.37	C#	88	8870
D	41	587.33	D	89	9397
D#	42	622.25	D#	90	9956
E	43	659.26	E	91	10548
F	44	698.46	F	92	11175
F#	45	739.99	F#	93	11840
G	46	783.99	G	94	12544
G#	47	830.61	G#	95	13290

3.3 Default Parameter Values

General Scanner	Header Definition	Default
Set Trigger Mode	GEN_PID_SET_TRIG_MODE	Level (0x01)
Picklist Mode	GEN_PID_PICKLIST_MODE	Disabled (0x00)
Scan Session Timeout	GEN_PID_SCAN_TIMEOUT	60 seconds in continuous mode; (0x3C) 10 seconds in single scan mode (0x0A)

Timeout between scans, same symbol	GEN_PID_TIMEOUT_BW_SAME_-SYM	0.6 second (0x06)
Timeout between scans, different symbols	GEN_PID_TIMEOUT_BW_DIFF_S-YM	0.2 second (0x02)
Continuous bar code read	GEN_PID_CONTINUOUS_READ	Enabled (0x01)
Unique bar code report	GEN_PID_UNIQUE_CODE_REPO-RT	Disabled (0x00)
Mobile Phone/Display Mode	GEN_PID_MOBILE_PHONE_MOD-E	Disabled (0x00)
Prefix key category	GEN_PID_PREFIX_KEY	1 (0x01)
Prefix Value	GEN_PID_PREFIX_VAL	<CR> (0x0D)
Suffix1 key category	GEN_PID_SUFFIX1_KEY	1(0x01)
Suffix1 Value	GEN_PID_SUFFIX1_VAL	<CR> (0x0D)
Suffix2 key category	GEN_PID_SUFFIX2_KEY	1 (0x01)
Suffix2 Value	GEN_PID_SUFFIX2_VAL	<CR> (0x0D)
Scan data transmission format	GEN_PID_SCAN_DATA_XMIT_F-MT	Data as is (0x01)
AIM pattern	GEN_PID_AIM_PATTERN_EN	Enabled (0x01)
Auto beep	GEN_PID_AUTO_BEEP_MODE	Disabled (0x00)
Scan beep	GEN_PID_AUTO_BEEP_SCAN	Frequency 164 msec, Duration 150 msec, Pause 50, Frequency 250 msec, Duration 2100 msec
Error beep	GEN_PID_AUTO_BEEP_ERROR	Frequency 158 msec, Duration 1100 msec, Pause75 msec, Frequency 250 msec, Duration 2100 msec.

Code 128	**Header Definition**	**Default**
Code 128	SYM_PID_EN_CODE128	Enable (0x01)
Set Length(s)	SYM_PID_SETLEN_X_C128	Any Length
GS1-128 (formerly UCC/EAN-128)	SYM_PID_EN_GS1128	Enable (0x01)
ISBT 128	SYM_PID_EN_ISBT	Enable (0x01)
ISBT concatenation	SYM_PID_ISBT_CONCATE	Disable (0x00)
Check ISBT table	SYM_PID_CHECK_ISBT_TABLE	Enable (0x01)
ISBT concatenation redundancy	SYM_PID_ISBT_CONCATE_RED-UN	10 (0x0A)

UPC/EAN	**Header Definition**	**Default**
UPC-A	SYM_PID_EN_UPCA	Enable (0x01)
UPC-E	SYM_PID_EN_UPCE	Enable (0x01)
UPC-E1	SYM_PID_EN_UPCE1	Disable (0x00)
EAN-8/JAN 8	SYM_PID_EN_EAN8_JAN8	Enable (0x01)
EAN-13/JAN 13	SYM_PID_EN_EAN13_JAN13	Enable (0x01)
Bookland EAN	SYM_PID_EN_BOOKLAND_EAN	Enable (0x01)
Bookland ISBN format	SYM_PID_EN_ISBN_FORMAT	ISBN-10 (0x00)
Decode UPC/EAN/JAN supplements	SYM_PID_SUPPLIMENTS_UPC_-EAN	Disable (0x00)
User-Programmable supplement1	SYM_PID_USER_PROG_SUPP1	N/A
User-Programmable supplement2 S	YM_PID_USER_PROG_SUPP2	N/A
UPC/EAN/JAN supplemental redundancy	SYM_PID_UPC_EAN_JAN_SUPP-_REDUN	10 (0x0A)

Transmit UPC-A Check Digit	SYM_PID_XMIT_UPCA_CHECK_-DIGIT	Enable (0x01)
Transmit UPC-E Check Digit	SYM_PID_XMIT_UPCE_CHECK_-DIGIT	Enable (0x01)
Transmit UPC-E1 Check Digit	SYM_PID_XMIT_UPCE1_CHECK-_DIGIT	Enable (0x01)
UPC-A Preamble	SYM_PID_XMIT_UPCA_PREAMB-LE	System Character (0x01)
UPC-E Preamble	SYM_PID_XMIT_UPCE_PREAMB-LE	System Character (0x01)
UPC-E1 Preamble	SYM_PID_XMIT_UPCE1_PREAM-BLE	System Character (0x01)
Convert UPC-E to A	SYM_PID_CONVERT_UPCE_2_U-PCA	Disable (0x00)
Convert UPC-E1 to A	SYM_PID_CONVERT_UPCE1_2_-UPCA	Disable (0x00)
EAN-8/JAN-8 Extend	SYM_PID_EAN8_JAN8_EXTEND	Disable (0x00)
UCC Coupon Extended Code	SYM_PID_UCC_COUPON_EXTE-ND	Disable (0x00)
Coupon Report	SYM_PID_COUPON_REPORT	New Coupon Symbols (0x01)
ISSN EAN	SYM_PID_ISSN_EAN	Disable (0x00)

Code 39	**Header Definition**	**Default**
Code 39	SYM_PID_EN_CODE39	Enable (0x01)
Trioptic Code 39	SYM_PID_EN_TRIOPTIC_COD-E39	Disable (0x00)
Convert Code 39 to Code 32	SYM_PID_CONV_CODE39_2_CO-DE32	Disable (0x00)
Code 32 Prefix	SYM_PID_CODE32_PREFIX	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_C39	Length range (0x020x37)
Code 39 Check Digit Verification	SYM_PID_CODE39_CHK_DIGIT	Disable (0x00)
Transmit Code 39 Check Digit	SYM_PID_XMIT_CODE39_CHK_-DIGIT	Disable (0x00)
Code 39 Full ASCII Conversion	SYM_PID_CODE39_FULL_ASCII	Disable (0x00)
Buffer Code 39	SYM_PID_CODE39_BUFFERING	Disable (0x00)

Interleaved 2 of 5	**Header Definition**	**Default**
Interleaved 2 of 5	SYM_PID_EN_INTER2OF5	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_I2OF5	One discrete length (0x0E)
I 2 of 5 Check Digit Verification	SYM_PID_I2OF5_CHECK_DIGIT	Disable (0x00)
Transmit I 2 of 5 Check Digit	SYM_PID_XMIT_I2OF5_CHECK_-DIGIT	Disable (0x00)
Convert I 2 of 5 to EAN 13	SYM_PID_CONV_I2OF5_EAN13	Disable (0x00)

QR Code	**Header Definition**	**Default**
QR Code	SYM_PID_EN_QR_CODE	Enable (0x01)
QR Inverse	SYM_PID_EN_QR_INVERSE	Regular (0x00)
MicroQR	SYM_PID_EN_MICRO_QR	Enable (0x01)

2D, Data Matrix	**Header Definition**	**Default**
Data Matrix	SYM_PID_EN_DATA_MATRIX	Enable (0x01)
Data Matrix Inverse	SYM_PID_EN_DATA_MATRIX_IN-VERSE	Regular (0x00)

Decode Mirror Images	SYM_PID_EN_MIRROR_IMAGES	Auto (0x02)

Code 93	Header Definition	Default
Code 93	SYM_PID_EN_CODE93	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_C93	Length range (0x040x37)

Code 11	Header Definition	Default
Code 11	SYM_PID_EN_CODE11	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_C11	Length range (0x04 0x37)
Code 11 check digit verification	SYM_PID_CODE11_CHK_DIGIT	Disable (0x00)
Transmit Code 11 Check Digit(s)	SYM_PID_XMIT_CODE11_CHK_-DIGIT	Disable (0x00)

Discrete 2 of 5	Header Definition	Default
Discrete 2 of 5	SYM_PID_EN_DISC2OF5	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_D2OF5	One discrete length (0x0C)

Codabar	Header Definition	Default
Codabar	SYM_PID_EN_CODABAR	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_CBAR	Length range (0x05 0x37)
CLSI Editing	SYM_PID_EN_CLSI	Disable (0x00)
NOTIS Editing	SYM_PID_EN_NOTIS	Disable (0x00)
Upper or Lower case Start/Stop char	SYM_PID_START_STOP_CASE	Upper Case (0x00)

MSI	Header Definition	Default
MSI	SYM_PID_EN_MSI	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_MSI	Length range (0x04 0x37)
MSI Check Digits	SYM_PID_MSI_CHK_DIGIT	One (0x00)
Transmit MSI Check Digit	SYM_PID_XMIT_MSI_CHK_DIGIT	Disable (0x00)
MSI Check Digit Algorithm	SYM_PID_MSI_CHK_DIGIT_ALG-OR	Mod 10/Mod 10 (0x00)

Chinese 2 of 5	Header Definition	Default
Chinese 2 of 5	SYM_PID_EN_CHINESE2OF5	Disable (0x00)

MATRIX 2 of 5	Header Definition	Default
Matrix 2 of 5	SYM_PID_EN_MATRIX2OF5	Disable (0x00)
Set Length(s)	SYM_PID_SETLEN_X_M2OF5	One discrete length (0x0E)
Matrix 2 of 5 Check Digit	SYM_PID_M2OF5_CHK_DIGIT	Disable (0x00)
Transmit Matrix 2 of 5 Check Digit	SYM_PID_XMIT_M2OF5_CHK_DI-GIT	Disable (0x00)

Korean 3 of 5	Header Definition	Default
Korean 3 of 5	SYM_PID_EN_KOREAN3OF5	Disable (0x00)

Inverse 1D	Header Definition	Default
Inverse 1D	SYM_PID_EN_INVERSE1D	Regular (0x00)

Postal Codes	Header Definition	Default
US Postnet	SYM_PID_EN_US_POSTNET	Disable (0x00)
US Planet	SYM_PID_EN_US_PLANET	Disable (0x00)
Transmit US Postal Check Digit	SYM_PID_XMIT_US_POST_CHK-_DIGIT	Enable (0x01)

UK Postal	SYM_PID_EN_UK_POSTAL	Disable (0x00)
Transmit UK Postal Check Digit	SYM_PID_XMIT_UK_POST_CHK-_DIGIT	Enable (0x01)
Japan Postal	SYM_PID_EN_JAPAN_POSTAL	Disable (0x00)
Australia Post	SYM_PID_EN_AUSTRALIA_POST	Disable (0x00)
Australia Post Format	SYM_PID_EN_AUST_POST_FMT	Auto discriminate (0x00)
Netherlands KIX Code	SYM_PID_EN_NETHERLANDS_K-IX	Disable (0x00)
USPS 4CB/One Code/Intelligent Mail	SYM_PID_EN_USPS_4CB	Disable (0x00)
UPU FICS Postal	SYM_PID_EN_UPU_FICS	Disable (0x00)

GS1 DataBar	Header Definition	Default
GS1 DataBar (omni directional, truncated,stacked, stacked omni directional)	SYM_PID_EN_GS1_DATABAR	Enable (0x01)
GS1 DataBar Limited	SYM_PID_EN_GS1_LIMITED	Disable (0x00)
GS1 DataBar Limited Security Level	SYM_PID_LTD_SECURITY	3 (0x03)
GS1 Databar Expanded	SYM_PID_EN_GS1_EXPANDED	Enable (0x01)
Convert GS1 DataBar to UPC/EAN	SYM_PID_EN_CONV_UPC_EAN	Disable (0x00)

Composite	Header Definition	Default
Composite CC-C	SYM_PID_EN_COMP_CC_C	Disable (0x00)
Composite CC-A/B	SYM_PID_EN_COMP_CC_A_B	Disable (0x00)
Composite TLC-39	SYM_PID_EN_COMP_TLC_39	Disable (0x00)
UPC Composite Mode	SYM_PID_UPC_COMP_MODE UPC	always linked (0x01)
GS1-128 Emulation Mode for UCC/EAN Composite codes	SYM_PID_EN_GS1_128_EMULAT-ION	Disable (0x00)

2D Barcodes	Header Definition	Default
PDF417	SYM_PID_EN_PDF417	Enable (0x01)
MicroPDF417	SYM_PID_EN_MICRO_PDF417	Disable (0x00)
Maxicode	SYM_PID_EN_MAXICODE	Disable (0x00)
Aztec	SYM_PID_EN_AZTEC	Enable (0x01)
Aztec Inverse	SYM_PID_EN_AZTEC_INVERSE	Inverse Auto detect (0x02)

3.4 Barcode_host.h reference file

```
/*****************************************************************************/
/* This source code is confidential proprietary information of VeriFone Inc. */
/* and is an unpublished work or authorship protected by the copyright laws  */
/* of the United States. Unauthorized copying or use of this document or the */
/* program contained herein, in original or modified form, is a violation of */
/* Federal and State law.                                                    */
/*                                                                           */
/*                           Verifone                                        */
/*                   1400 West Stanford Ranch Road                           */
/*                      Rocklin, CA 95765                                    */
/*                        Copyright 2013                                     */
/*****************************************************************************/
```

```
/*
 ;|=========================================================================+
 ;| FILE NAME:  | Barcode_host.h
 ;|-------------+-----------------------------------------------------------+
 ;| DEVICE:     | Any Barcode Host device
 ;|-------------+-----------------------------------------------------------+
 ;| DESCRIPTION:|Host interface defines for symbologies and host command packet
 ;|             |definitions.
 ;|-------------+-----------------------------------------------------------+
 ;| TYPE:       |
 ;|-------------+-----------------------------------------------------------+
 ;| COMMENTS:   |
 ;|=========================================================================+
 */

#define HOST_BUF_SIZE  2048

// Beeper command defintions
typedef struct {
    int freq;
    int dur;
}T_BEEP;

typedef struct {
    T_BEEP b1;
    int bPause;
    T_BEEP b2;
}T_BEEP_DEF;

typedef struct {
    int freq_1;
    int duration_1;
    int pause_1;
    int freq_2;
    int duration_2;
    int pause_2;
    int freq_3;
    int duration_3;
    int pause_3;
```

```
    int freq_4;

    int duration_4;

    int pause_4;

    int freq_5;

    int duration_5;

    int pause_5;

    int freq_6;

    int duration_6;

    int pause_6;

}M_BEEP_DEF;


// This Structure is the parameter buffer definition

typedef struct {

            unsigned char  length;                                                                  //parameter
       length

            unsigned char  pId;
       //parameter ID number (any value from defines starting with SYM_PID or GEN)

    //      union {

    //        char bytes[HOST_BUF_SIZE-14];                                        //variable number of
       parameter values depends on the length

    //        T_BEEP_DEF beep;

    //      } pValue;

}PARM_DEF;


// This Structure is the command buffer definition

typedef struct {

    unsigned short int length;                  // command packet length

    unsigned char protoVer;                     // protocol version

            unsigned char command;                                      // command number (any value from
       defines starting with CMD)

    //    char params[HOST_BUF_SIZE-12];                                // variable number of parameters
       depends on the length

}CMD_DEF;


// Barcode Application Command format (host device to e210, e255 and e315 Barcode Application)

// Commands from Host Interface to Barcode App use this command structure

typedef struct {

    unsigned long length;                        // total length of the host packet (max value 2048 bytes)

    unsigned long sequence;                      // host command sequence number (serial number assigned by
       host)

    //    char cmnds[HOST_BUF_SIZE-8];                 // variable number of commands depends on the length

}HOST_TX_BUF;
```

```
// Barcode Application Response format (Barcode Application to Host Device)

typedef struct {

    unsigned long length;                                                                    //total
       length of the response packet

    unsigned long sequence;                                                                  //response
       packet sequence number

         unsigned long status;                                                               //last
       received packet process status, 0 - success in processing whole packet, 1 - failure

    //    char data[HOST_BUF_SIZE-12];                                          //optional data

}HOST_RX_BUF;



// Host command definitions

// These are commands between host device and Verix application



#define CMD_BAR_DEV_OPEN 0x1A

#define CMD_BAR_DEV_CLOSE 0x1B

#define CMD_START_SCAN 0x01

#define CMD_STOP_SCAN 0x02

#define CMD_PASS_THRU 0x03

#define CMD_SINGLE_GET_APP_VER 0x04

#define CMD_SET_TRIG_MODE 0x05

#define CMD_BEEP_IMMEDIATE 0x06

#define CMD_SINGLE_BUTTON_STATUS 0x07

#define CMD_SINGLE_GET_FIRM_VER 0x08

#define CMD_SINGLE_GET_DEVICE_ID 0x09

#define CMD_MULTI_SCAN 0x0A

#define CMD_RESTORE_DEFAULTS 0x0B

#define CMD_EN_PICKLIST_MODE 0x0C

#define CMD_SCAN_TIMEOUT 0x0D

#define CMD_TIMEOUT_BW_SAME_SYM 0x0E

#define CMD_TIMEOUT_BW_DIFF_SYM 0x0F

#define CMD_EN_CONTINUOUS_RD 0x10

#define CMD_EN_UNIQUE_CODE_REPO 0x11

#define CMD_EN_MOBILE_PH_MODE 0x12

#define CMD_PREFIX_VAL 0x13

#define CMD_SUFFIX1_VAL 0x14

#define CMD_SUFFIX2_VAL 0x15

#define CMD_EN_XMIT_FMT 0x16

#define CMD_EN_AIM_PATTERN 0x17

#define CMD_SYMBOLOGY 0x18

#define CMD_DISABLE_ALL_SYMB 0x19
```

```
#define CMD_AUTO_BEEP_CONFIG 0x1C
#define CMD_GET_AUTO_BEEP_CONFIG 0x1D


// parameter identifier table definitions

//code 128
#define SYM_PID_EN_CODE128 0x00 //value 1 for *enable, 0 disable
#define SYM_PID_SETLEN_1DISCRETE_C128 0x01 //one byte input value
#define SYM_PID_SETLEN_2DISCRETE_C128 0x02 //two bytes input value
#define SYM_PID_SETLEN_RANGE_C128 0x03 //Range is set by two byte input value
#define SYM_PID_SETLEN_ANY_C128 0x04 //*No input value expected for this parameter
#define SYM_PID_EN_GS1128 0x05 //value 1 for *enable, 0 disable
#define SYM_PID_EN_ISBT 0x06 //value 1 for *enable, 0 disable
#define SYM_PID_ISBT_CONCATE 0x07 //value 1 for enable, 0 *disable, 2 auto
#define SYM_PID_CHECK_ISBT_TABLE 0x08 //value 1 for *enable, 0 disable
#define SYM_PID_ISBT_CONCATE_REDUN 0x09 //value range is 2 to 20, *0x0A
//code UPC
#define SYM_PID_EN_UPCA 0x0A //value 1 for *enable, 0 disable
#define SYM_PID_EN_UPCE 0x0B //value 1 for *enable, 0 disable
#define SYM_PID_EN_UPCE1 0x0C //value 1 for enable, 0 *disable
#define SYM_PID_EN_EAN8_JAN8 0x0D //value 1 for *enable, 0 disable
#define SYM_PID_EN_EAN13_JAN13 0x0E //value 1 for *enable, 0 disable
#define SYM_PID_EN_BOOKLAND_EAN 0x0F //value 1 for *enable, 0 disable
#define SYM_PID_EN_ISBN_FORMAT 0x10 //value 1 for enabling ISBN-13 & 0 *ISBN-10
#define SYM_PID_SUPPLIMENTS_UPC_EAN 0x11 //possible values of this parameter are given below
#define IGNORE_UPC_EAN_W_SUPPLIMENTS *0x00
#define DECODE_UPC_EAN_W_SUPPLIMENTS 0x01
#define AUTODESCRIMINATE_SUPPLIMENTS 0x02
#define ENABLE_378_379_SUPPLIMENTS 0x04
#define ENABLE_978_979_SUPPLIMENTS 0x05
#define ENABLE_977_SUPPLIMENT 0x07
#define ENABLE_414_419_434_439 0x06
#define ENABLE_491_SUPPLIMENTS 0x08
#define ENABLE_SMART_SUPPLIMENT 0x03
#define SUPPLIMENT_USER_PROG1 0x09
#define SUPPLIMENT_USER_PROG1_2 0x0A
#define SMART_SUPPLIMENT_USER_PROG1 0x0B
#define SMART_SUPPLIMENT_USER_PROG1_2 0x0C
#define SYM_PID_USER_PROG_SUPP1 0x12 //3 digit value input
#define SYM_PID_USER_PROG_SUPP2 0x13 //3 digit value input
```

```
#define SYM_PID_UPC_EAN_JAN_SUPP_REDUN 0x14 //range is 2 to 30, *0x0A

#define SYM_PID_XMIT_UPCA_CHECK_DIGIT 0x15 //value 1 for *enable,0 disable

#define SYM_PID_XMIT_UPCE_CHECK_DIGIT 0x16 //value 1 for *enable,0 disable

#define SYM_PID_XMIT_UPCE1_CHECK_DIGIT 0x17 //value 1 for *enable,0 disable

#define SYM_PID_XMIT_UPCA_PREAMBLE 0x18 //value 0 for No Preamble,1 *system char & 2 country code & system
       char

#define SYM_PID_XMIT_UPCE_PREAMBLE 0x19 //value 0 for No Preamble,1 *system char & 2 country code & system
       char

#define SYM_PID_XMIT_UPCE1_PREAMBLE 0x1A //value 0 for No Preamble,1 *system char & 2 country code & system
       char

#define SYM_PID_CONVERT_UPCE_2_UPCA 0x1B //value 1 for enable,0 *disable

#define SYM_PID_CONVERT_UPCE1_2_UPCA 0x1C //value 1 for enable,0 *disable

#define SYM_PID_EAN8_JAN8_EXTEND 0x1D //value 1 for enable,0 *disable

#define SYM_PID_UCC_COUPON_EXTEND 0x1E //value 1 for enable,0 *disable

#define SYM_PID_COUPON_REPORT 0x1F //value 0 for old coupon symbols,1 *new coupon symbol,2 both coupons

#define SYM_PID_ISSN_EAN 0x20 //value 1 for enable,0 disable

//code 39

#define SYM_PID_EN_CODE39 0x21 //value 1 for *enable,0 disable

#define SYM_PID_EN_TRIOPTIC_CODE39 0x22 //value 1 for enable,0 *disable

#define SYM_PID_CONV_CODE39_2_CODE32 0x23 //value 1 for enable,0 *disable

#define SYM_PID_CODE32_PREFIX 0x24 //value 1 for enable,0 *disable

#define SYM_PID_SETLEN_1DISCRETE_C39 0x25 //one byte input value

#define SYM_PID_SETLEN_2DISCRETE_C39 0x26 //two bytes input value

#define SYM_PID_SETLEN_RANGE_C39 0x27 //*Range is set by two byte input value,*0x02-0x37

#define SYM_PID_SETLEN_ANY_C39 0x28 //No input value expected for this parameter

#define SYM_PID_CODE39_CHK_DIGIT 0x29 //value 1 for enable,0 *disable

#define SYM_PID_XMIT_CODE39_CHK_DIGIT 0x2A //value 1 for enable,0 *disable

#define SYM_PID_CODE39_FULL_ASCII 0x2B //value 1 for enable,0 *disable

#define SYM_PID_CODE39_BUFFERING 0x2C //value 1 for enable,0 *disable

//Interleaved 2 of 5

#define SYM_PID_EN_INTER2OF5 0x2D //value 1 for enable,0 *disable

#define SYM_PID_SETLEN_1DISCRETE_I2OF5 0x2E //*one byte input value, *0x0E

#define SYM_PID_SETLEN_2DISCRETE_I2OF5 0x2F //two bytes input value

#define SYM_PID_SETLEN_RANGE_I2OF5 0x30 //Range is set by two byte input value

#define SYM_PID_SETLEN_ANY_I2OF5 0x31 //No input value expected for this parameter

#define SYM_PID_I2OF5_CHECK_DIGIT 0x32 //value 1 for enable,0 *disable

#define SYM_PID_XMIT_I2OF5_CHECK_DIGIT 0x33 //value 1 for enable,0 *disable EXAMPLE HEADER FILE 44 e255
       BARCODE APPLICATION PROGRAMMERS GUIDE

#define SYM_PID_CONV_I2OF5_EAN13 0x34 //value 1 for enable,0 *disable

//2D,QR code

#define SYM_PID_EN_QR_CODE 0x35 //value 1 for *enable,0 disable

//2D,QR Inverse

#define SYM_PID_EN_QR_INVERSE 0x36 //value 0 *regular,1 Inverse,2 Auto
```

```
//2D, MicroQR
#define SYM_PID_EN_MICRO_QR 0x37 //value 1 for *enable,0 disable
//2D,Data Matrix
#define SYM_PID_EN_DATA_MATRIX 0x38 //value 1 for *enable,0 disable
#define SYM_PID_EN_DATA_MATRIX_INVERSE 0x39 //value 0 *regular,1 Inverse,2 Auto
#define SYM_PID_EN_MIRROR_IMAGES 0x3A //value 0 regular,1 Inverse, 2 *Auto
//Code 93
#define SYM_PID_EN_CODE93 0x3B //value 1 for enable, 0 *disable
#define SYM_PID_SETLEN_1DISCRETE_C93 0x3C //one byte input value
#define SYM_PID_SETLEN_2DISCRETE_C93 0x3D //two bytes input value
#define SYM_PID_SETLEN_RANGE_C93 0x3E //*Range is set by two byte input value, *0x04-0x37
#define SYM_PID_SETLEN_ANY_C93 0x3F //No input value expected for this parameter
//Code 11
#define SYM_PID_EN_CODE11 0x40 //value 1 for enable,0 *disable
#define SYM_PID_SETLEN_1DISCRETE_C11 0x41 //one byte input value
#define SYM_PID_SETLEN_2DISCRETE_C11 0x42 //two bytes input value
#define SYM_PID_SETLEN_RANGE_C11 0x43 //Range is set by two byte input value,*0x04-0x37
#define SYM_PID_SETLEN_ANY_C11 0x44 //No input value expected for this parameter
#define SYM_PID_CODE11_CHK_DIGIT 0x45 //value 0 for *disable,1 one check digit & 2 two check digits
#define SYM_PID_XMIT_CODE11_CHK_DIGIT 0x46 //value 1 for enable,0 *disable
//Discrete 2 of 5
#define SYM_PID_EN_DISC2OF5 0x47 //value 1 for enable,0 *disable
#define SYM_PID_SETLEN_1DISCRETE_D2OF5 0x48 //*one byte input value, 0x0C
#define SYM_PID_SETLEN_2DISCRETE_D2OF5 0x49 //two bytes input value
#define SYM_PID_SETLEN_RANGE_D2OF5 0x4A //Range is set by two byte input value
#define SYM_PID_SETLEN_ANY_D2OF5 0x4B //No input value expected for this parameter
//Codabar
#define SYM_PID_EN_CODABAR 0x4C //value 1 for enable,0 disable
#define SYM_PID_SETLEN_1DISCRETE_CBAR 0x4D //one byte input value
#define SYM_PID_SETLEN_2DISCRETE_CBAR 0x4E //two bytes input value
#define SYM_PID_SETLEN_RANGE_CBAR 0x4F //*Range is set by two byte input value,*0x05-0x37
#define SYM_PID_SETLEN_ANY_CBAR 0x50 //No input value expected for this parameter
#define SYM_PID_EN_CLSI 0x51 //value 1 for enable,0 *disable
#define SYM_PID_EN_NOTIS 0x52 //value 1 for enable,0 *disable EXAMPLE HEADER FILE e255 BARCODE APPLICATION
       PROGRAMMERS GUIDE 45
#define SYM_PID_START_STOP_CASE 0x53 //value 1 for lower case,0 *upper case
//MSI
#define SYM_PID_EN_MSI 0x54 //value 1 for enable,0 *disable
#define SYM_PID_SETLEN_1DISCRETE_MSI 0x55 //one byte input value
#define SYM_PID_SETLEN_2DISCRETE_MSI 0x56 //two bytes input value
#define SYM_PID_SETLEN_RANGE_MSI 0x57 //*Range is set by two byte input value,*0x04-0x37
```

```
#define SYM_PID_SETLEN_ANY_MSI 0x58 //No input value expected for this parameter

#define SYM_PID_MSI_CHK_DIGIT 0x59 //value 0 for *one check digit,1 two check digits

#define SYM_PID_XMIT_MSI_CHK_DIGIT 0x5A //value 1 for enable,0 *disable

#define SYM_PID_MSI_CHK_DIGIT_ALGOR 0x5B //value 0 for MOD 10/11 algorithm,1 *MOD10/MOD10 algorithm

//CHINESE 2 of 5

#define SYM_PID_EN_CHINESE2OF5 0x5C //value 1 for enable,0 *disable

//MATRIX 2 of 5

#define SYM_PID_EN_MATRIX2OF5 0x5D //value 1 for enable,0 *disable

#define SYM_PID_SETLEN_1DISCRETE_M2OF5 0x5E //*one byte input value,*0x0E

#define SYM_PID_SETLEN_2DISCRETE_M2OF5 0x5F //two bytes input value

#define SYM_PID_SETLEN_RANGE_M2OF5 0x60 //Range is set by two byte input value

#define SYM_PID_SETLEN_ANY_M2OF5 0x61 //No input value expected for this parameter

#define SYM_PID_M2OF5_CHK_DIGIT 0x62 //value 1 for enable,0 *disable

#define SYM_PID_XMIT_M2OF5_CHK_DIGIT 0x63 //value 1 for enable,0 *disable

//KOREAN 3 of 5

#define SYM_PID_EN_KOREAN3OF5 0x64 //value 1 for enable,0 *disable

//INVERSE 1D

#define SYM_PID_EN_INVERSE1D 0x65 //value 0 for *regular,1 inverse & 2 inverse auto detect

//POSTAL CODES

#define SYM_PID_EN_US_POSTNET 0x66 //value 1 for enable,0 *disable

#define SYM_PID_EN_US_PLANET 0x67 //value 1 for enable,0 *disable

#define SYM_PID_XMIT_US_POST_CHK_DIGIT 0x68 //value 1 for *enable,0 disable

#define SYM_PID_EN_UK_POSTAL 0x69 //value 1 for enable,0 *disable

#define SYM_PID_XMIT_UK_POST_CHK_DIGIT 0x6A //value 1 for *enable,0 disable

#define SYM_PID_EN_JAPAN_POSTAL 0x6B //value 1 for enable,0 *disable

#define SYM_PID_EN_AUSTRALIA_POST 0x6C //value 1 for enable,0 *disable

#define SYM_PID_EN_AUST_POST_FMT 0x6D //value 0 for *auto,1 raw format,2 alphanum enc,3 num enc

#define SYM_PID_EN_NETHERLANDS_KIX 0x6E //value 1 for enable,0 *disable

#define SYM_PID_EN_USPS_4CB 0x6F //value 1 for enable,0 *disable

#define SYM_PID_EN_UPU_FICS 0x70 //value 1 for enable,0 *disable

//GS1 DataBar

#define SYM_PID_EN_GS1_DATABAR 0x71 //value 1 for *enable,0 disable

#define SYM_PID_EN_GS1_LIMITED 0x72 //value 1 for enable,0 *disable

#define SYM_PID_LTD_SECURITY 0x73 //value 1 for security level 1,2 level2,3 *level3,4 level4 EXAMPLE HEADER
       FILE 46 e255 BARCODE APPLICATION PROGRAMMERS GUIDE

#define SYM_PID_EN_GS1_EXPANDED 0x74 //value 1 for *enable,0 disable

#define SYM_PID_EN_CONV_UPC_EAN 0x75 //value 1 for enable,0 *disable

//COMPOSITE

#define SYM_PID_EN_COMP_CC_C 0x76 //value 1 for enable,0 *disable

#define SYM_PID_EN_COMP_CC_A_B 0x77 //value 1 for enable,0 *disable

#define SYM_PID_EN_COMP_TLC_39 0x78 //value 1 for enable,0 *disable
```

```
#define SYM_PID_UPC_COMP_MODE 0x79 //value 0 for UPC never linked, 1 *always linked,2 auto

#define SYM_PID_EN_GS1_128_EMULATION 0x7A //value 1 for enable,0 *disable

//2D, PDF417

#define SYM_PID_EN_PDF417 0x7B //value 1 for *enable,0 disable

//2D, MICRO PDF417

#define SYM_PID_EN_MICRO_PDF417 0x7C //value 1 for enable,0 *disable

//2D, Maxicode

#define SYM_PID_EN_MAXICODE 0x7D //value 1 for enable,0 *disable

//2D, Aztec

#define SYM_PID_EN_AZTEC 0x7E //value 1 for *enable,0 disable

//2D, Aztec Inverse

#define SYM_PID_EN_AZTEC_INVERSE 0x7F //value 0 for regular,1 inverse, 2 *inverse auto detect

//general parameter IDs

// asterick (*) in the comments indicate default value

#define GEN_PID_PASS_THRU 0xFF //reserved value

#define GEN_PID_SET_TRIG_MODE 0xFE //8 bit value,0 edge,1 *level,2 soft, 3 passive

#define GEN_PID_BEEP_IMMEDIATE 0xFD //value starts with no of beeps followed no. of T_BEEP_PAUSE

#define GEN_PID_RESTORE_DEFAULTS 0xFC //value is not required and reserved for future

#define GEN_PID_PICKLIST_MODE 0xFB //value 0 *disabled,1 enabled

#define GEN_PID_SCAN_TIMEOUT 0xFA //value range 1 to 255 secs,*60 sec in continuous,*10 sec in single scan

#define GEN_PID_TIMEOUT_BW_SAME_SYM 0xF9 //value range 0 to 9.9 sec (99 decimal), *0.6 sec(0x06)

#define GEN_PID_TIMEOUT_BW_DIFF_SYM 0xF8 //value range 1 to 9.9 sec (99 decimal), *0.2 sec(0x02)

#define GEN_PID_CONTINUOUS_READ 0xF7 //value 0 disabled,1 *enabled

#define GEN_PID_UNIQUE_CODE_REPORT 0xF6 //value 0 *disabled,1 enabled

#define GEN_PID_MOBILE_PHONE_MODE 0xF5 //value 0 *disabled,1 enabled

#define GEN_PID_PREFIX_KEY 0xF4 //value is 1 when host sends prefix value

#define GEN_PID_PREFIX_VAL 0xF3 //value any 3 digit number 0-255,*<CR>(0x0D)

#define GEN_PID_SUFFIX1_KEY 0xF2 //value is 1 when host sends prefix value

#define GEN_PID_SUFFIX1_VAL 0xF1 //value any 3 digit number 0-255,*<CR>(0x0D)

#define GEN_PID_SUFFIX2_KEY 0xF0 //value is 1 when host sends prefix value EXAMPLE HEADER FILE e255 BARCODE
       APPLICATION PROGRAMMERS GUIDE 47

#define GEN_PID_SUFFIX2_VAL 0xEF //value any 3 digit number 0-255,*<CR>(0x0D)

#define GEN_PID_SCAN_DATA_XMIT_FMT 0xEE //value 0 *data as is, 1 sufix1, 2 sufix2,3 sufix1&2,4 prefix, 5
       Prefix&sufix1,6 prefix&sufix2, 7 prefix&suffixes

#define GEN_PID_AIM_PATTERN_EN 0xED //value 0 disabled,1 *enabled

#define GEN_PID_AUTO_BEEP_MODE 0xEC //value 0 *disabled,1 config scan beep only,2 config error beep only, 3
       config both

#define GEN_PID_AUTO_BEEP_SCAN 0xEB //scan beep value in format T_BEEP_DEF

#define GEN_PID_AUTO_BEEP_ERROR 0xEA //error beep value in format T_BEEP_DEF
```

```
// Host response definitions
// This is a response from Verix application to host device

#define CMD_BARCODE_DATA                        0x01




// Verix device response to Host

#define RESP_ACK 0x00000000
#define RESP_NAK 0x00000001
#define RESP_BARCODE_DATA 0x80000000
#define RESP_BUTTON_STATUS 0x80000001


// Barcode data header

typedef struct {
    char codeID;
    char AIMID;
    unsigned short int symbology;
}SYMB_INFO;


// Code ID is one of the industry standard and parameter changes will refer to a specific code ID.
// each code ID has an associated code ASCII character that needs to be displayed in the retail
       environment.
// Refer to symb_code array for asscociation between code ID and code ASCII char.
// Every succesfull decoded data will have a unique symbology as one of the return values

#define CODE_ID_UPC_EA 0x01
#define CODE_ID_CODE39_32 0x02
#define CODE_ID_CODABAR 0x03
#define CODE_ID_CODE128_ISBT 0x04
#define CODE_ID_CODE93 0x05
#define CODE_ID_INTL2OF5 0x06
#define CODE_ID_DISC2OF5 0x07
#define CODE_ID_CODE11 0x08
#define CODE_ID_MSI 0x09
#define CODE_ID_GSI128 0x0A
#define CODE_ID_BOOKLAND_EAN 0x0B
#define CODE_ID_TRIOPTIC39 0x0C
#define CODE_ID_COUPONCODE 0x0D
#define CODE_ID_GS1DATABAR 0x0E
```

```
#define CODE_ID_MATRIX2OF5 0x0F
#define CODE_ID_UCCCOMPOS 0x10
#define CODE_ID_CHINESE2OF5 0x11
#define CODE_ID_KOREAN3OF5 0x12
#define CODE_ID_PDF417_ISSNEAN 0x13
#define CODE_ID_AZTEC_RUNE 0x14
#define CODE_ID_DATA_MATRIX 0x15
#define CODE_ID_QRCODE_MICRO 0x16
#define CODE_ID_MAXICODE 0x17
#define CODE_ID_US_POSTNET 0x18
#define CODE_ID_US_PLANET 0x19
#define CODE_ID_JAPAN_POSTAL 0x1A
#define CODE_ID_UK_POSTAL 0x1B
#define CODE_ID_POSTBAR_CA 0x1C
#define CODE_ID_NETH_KIX 0x1D
#define CODE_ID_AUS_POST 0x1E
#define CODE_ID_USPS_4CB 0x1F
#define CODE_ID_UPU_FICS 0x20
#define CODE_ID_SCANLET_WEB 0x21
#define CODE_ID_CUECAT 0x22


//symbology code array, indexed by code ID defines


const char* symb_code [] = {
    "NA", //0x00
    "A", //UPC/EAN //0x01
    "B", //Code 39,Code 32 //0x02
    "C", //Codabar //0x03
    "D", //Code 128, ISBT 12 //0x04
    "E", //Code 93 //0x05
    "F", //Interleaved 2 of 5 //0x06
    "G", //Discrete 2 of 5 //0x07
    "H", //Code 11 //0x08
    "J", //MSI //0x09
    "K", //GS1-128 //0x0A
    "L", //Bookland EAN //0x0B
    "M", //Trioptic Code 39 //0x0C
```

```
    "N", //Coupon Code //0x0D
    "R", //GS1 DataBar Family //0x0E
    "S", //Matrix 2 of 5 //0x0F
    "T", //UCC Composite, TLC 39 //0x10
    "U", //Chinese 2 of 5 //0x11
    "V", //Korean 3 of 5 //0x12
    "X", //ISSN EAN,PDF417,MacroPDF417 //0x13
    "z", //Aztec, Azec Rune //0x14
    "P00", //Data Matrix //0x15
    "P01", //QR Code, MicroQR //0x16
    "P02", //Maxicode //0x17
    "P03", //US Postnet //0x18
    "P04", //US Planet //0x19
    "P05", //Japan Postal //0x1A
    "P06", //UK Postal //0x1B
    "P07", //Postbar CA //0x1C
    "P08", //Netherlands KIX code //0x1D
    "P09", //Australia Post //0x1E
    "P0A", //USPS 4CB //0x1F
    "P0B", //UPU FICS Postal //0x20
    "W", //scanlet webcode //0x21
    "Q "//Cue CAT code //0x22
};


// AIM ID is altrenate industry standard and only used for display purpose.
// each AIM ID has an associated AIM ASCII character that needs to be displayed in the retail environment.
// Refer to symb_AIM array for asscociation between AIM ID and AIM ASCII char.
// Every succesfull decoded data will have a unique symbology as one of the return values

#define AIM_ID_CODE39_32 0x01
#define AIM_ID_CODE128_ISBT_GS1 0x02
#define AIM_ID_DATAMATRIX 0x03
#define AIM_ID_UPC_EAN_COUPON 0x04
#define AIM_ID_GS1DATABAR 0x05
#define AIM_ID_CODEBAR 0x06
#define AIM_ID_CODE93 0x07
#define AIM_ID_CODE11 0x08
#define AIM_ID_INTL2OF5 0x09
#define AIM_ID_PDF417 0x0A
#define AIM_ID_TLC39 0x0B
#define AIM_ID_MSI 0x0C
```

```
#define AIM_ID_QRCODE_MICROQR 0x0D
#define AIM_ID_DISC2OF5 0x0E
#define AIM_ID_MAXICODE 0x0F
#define AIM_ID_AZTEC_RUNE 0x10
#define AIM_ID_X 0x11
#define AIM_ID_COMP_EC 0x12
#define AIM_ID_COMP_EE 0x13
#define AIM_ID_COMP_RS 0x14



//symbology AIM array, indexed by AIM ID defines


extern const char* symb_AIM[];


const char* symb_AIM [] = {
    "NA",
    "A",
    "C",
    "d",
    "E",
    "e",
    "F",
    "G",
    "H",
    "I",
    "L",
    "L2",
    "M",
    "Q",
    "S",
    "U",
    "z",
    "X",
    "E+C",
    "E+E",
    "RS "
};


// 16-bit symbology values in barcode data response
```

```
#define SYM_ID_CODE39 0x0001
#define SYM_ID_CODABAR 0x0002
#define SYM_ID_CODE128 0x0003
#define SYM_ID_D25 0x0004
#define SYM_ID_IATA 0x0005
#define SYM_ID_ITF 0x0006
#define SYM_ID_CODE93 0x0007
#define SYM_ID_UPCA 0x0008
#define SYM_ID_UPCE 0x0009
#define SYM_ID_EAN8 0x000A
#define SYM_ID_EAN13 0x000B
#define SYM_ID_CODE11 0x000C
#define SYM_ID_MSI 0x000D
#define SYM_ID_EAN128 0x000E
#define SYM_ID_UPCE1 0x000F
#define SYM_ID_PDF417 0x0010
#define SYM_ID_CODE39FULL 0x0011
#define SYM_ID_TRIOPTIC 0x0012
#define SYM_ID_BOOKLAND 0x0013
#define SYM_ID_COUPONCODE 0x0014
#define SYM_ID_ISBT128 0x0015
#define SYM_ID_MICROPDF 0x0016
#define SYM_ID_DATAMATRIX 0x0017
#define SYM_ID_QRCODE 0x0018
#define SYM_ID_POSTNETUS 0x0019
#define SYM_ID_PLANETUS 0x001A
#define SYM_ID_CODE32 0x001B
#define SYM_ID_ISBT128CONC 0x001C
#define SYM_ID_POSTALJAPAN 0x001D
#define SYM_ID_POSTALAUST 0x001E
#define SYM_ID_POSTALDUTCH 0x001F
#define SYM_ID_MAXICODE 0x0020
#define SYM_ID_POSTBARCA 0x0021
#define SYM_ID_POSTALUK 0x0022
#define SYM_ID_MACROPDF417 0x0023
#define SYM_ID_RSS14 0x0024
#define SYM_ID_RSSLIMIT 0x0025
#define SYM_ID_RSSEXPAND 0x0026
#define SYM_ID_SCANLETWEB 0x0027
#define SYM_ID_CUECAT 0x0028
```

```
#define SYM_ID_UPCA_2 0x0029
#define SYM_ID_UPCE_2 0x002A
#define SYM_ID_EAN8_2 0x002B
#define SYM_ID_EAN13_2 0x002C
#define SYM_ID_UPCE1_2 0x002D
#define SYM_ID_CCA_EAN128 0x002E
#define SYM_ID_CCA_EAN13 0x002F
#define SYM_ID_CCA_EAN8 0x0030
#define SYM_ID_CCA_RSSEXPAND 0x0031
#define SYM_ID_CCA_RSSLIMIT 0x0032
#define SYM_ID_CCA_RSS14 0x0033
#define SYM_ID_CCA_UPCA 0x0034
#define SYM_ID_CCA_UPCE 0x0035
#define SYM_ID_CCC_EAN128 0x0036
#define SYM_ID_TLC39 0x0037
#define SYM_ID_CCB_EAN128 0x0038
#define SYM_ID_CCB_EAN13 0x0039
#define SYM_ID_CCB_EAN8 0x003A
#define SYM_ID_CCB_RSSEXPAND 0x003B
#define SYM_ID_CCB_RSSLIMIT 0x003C
#define SYM_ID_CCB_RSS14 0x003D
#define SYM_ID_CCB_UPCA 0x003E
#define SYM_ID_CCB_UPCE 0x003F
#define SYM_ID_KOR3OF5 0x0040
#define SYM_ID_UPCA_5 0x0041
#define SYM_ID_UPCE_5 0x0042
#define SYM_ID_EAN8_5 0x0043
#define SYM_ID_EAN13_5 0x0044
#define SYM_ID_UPCE1_5 0x0045
#define SYM_ID_MACROPDF 0x0046



//look-up table to retrieve symbology names

const char* symbology[] = {
    "NA",
    "Code 39",
    "Codabar",
    "Code 128",
    "D25",
```

```
"IATA",
"ITF",
"Code 93",
"UPCA",
"UPCE",
"EAN-8",
"EAN-13",
"Code 11",
"MSI",
"EAN-128",
"UPCE1",
"PDF-417",
"Code 39 FULL ASCII",
"Trioptic",
"Bookland",
"Coupon Code",
"ISBT-128",
"Micro PDF",
"Data Matrix",
"QR Code",
"PostnetUS",
"PlanetUS",
"Code 32",
"ISBT-128 Concat",
"PostalJapan",
"PostalAustralia",
"PostalDutch",
"Maxicode",
"PostbarCA",
"PostalUK",
"MacroPDF-417",
"RSS-14",
"RSS Limited",
"RSS Expanded",
"ScanletWebcode",
"CueCATCode",
"UPCA+2",
"UPCE+2",
"EAN8+2",
"EAN13+2",
"UPCE1+2",
```

```
    "CompositeCCA+EAN128",
    "CompositeCCA+EAN13",
    "CompositeCCA+EAN8",
    "CompositeCCA+RSSExpand",
    "CompositeCCA+RSSLimit",
    "CompositeCCA+RSS14",
    "CompositeCCA+UPCA",
    "CompositeCCA+UPCE",
    "CompositeCCC+EAN128",
    "TLC39",
    "CompositeCCB+EAN128",
    "CompositeCCB+EAN13",
    "CompositeCCB+EAN8",
    "CompositeCCB+RSSExpand"
    "CompositeCCB+RSSLimit",
    "CompositeCCB+RSS14",
    "CompositeCCB+UPCA",
    "CompositeCCB+UPCE",
    "KOREAN3of5",
    "UPCA+5",
    "UPCE+5",
    "EAN8+5",
    "EAN13+5",
    "UPCE1+5",
    "MacroMicroPDF"
};
// parameter standard values

#define DISABLE 0x00
#define ENABLE          0x01
```

Chapter 4

VCMF API Reference Guide

The VeriFone Mobile Communication Framework (VCMF) is an iPhone Framework that will be provided by VeriFone as the main interface between iPhone applications wishing to utilize the e210/e255/e315 payment device for facilitating the capture of customer information (track data, PIN, EMV chip, etc.) necessary to process payments with the iPhone applications designated host processor. The VM- Framework will provide access to the individual XPI and barcode command set as well as provide for more convenient flow function commands in order to simplify the burden on the application to know the complete series of commands necessary to implement a fully-featured payment flow with the customer.

The purpose of this document is to describe the requirements of the VM-Framework as well as the interface definitions and requirements needed for any iPhone applications wishing to deploy with the payment framework.

- Important Security Notice
- Introduction
- Installation and Integration Setup
- VFIConnect Implementation
- Unique Serial Number Requirement

4.1 Important Security Notice

The Payment Card Industry Payment Application Data Security Standard (PCI PA-DSS) is comprised of fourteen requirements that support the Payment Card Industry Data Security Standard (PCI DSS). The PCI Security Standards Council (PCI SSC), which was founded by the major card brands in June 2005, set these requirements in order to protect cardholder payment information. The standards set by the council are enforced by the payment card companies who established the Council: American Express, Discover Financial Services, JCB International, MasterCard Worldwide, and Visa, Inc.

PCI PA-DSS is an evolution of Visas Payment Application Best Practices (PABP), which was based on the Visa Cardholder Information Security Program (CISP). In addition to Visa CISP, PCI DSS combines American Express Data Security Operating Policy (DSOP), Discover Networks Information Security and Compliance (DISC), and MasterCards

Site Data Protection (SDP) into a single comprehensive set of security standards. The transition to PCI PA-DSS was announced in April 2008. In early October 2008, PCI PA-DSS Version 1.2 was released to align with the PCI DSS Version 1.2, which was released on October 1, 2008. On January 1, 2011, PCI PA-DSS Version 2.0 was released. This extends the PCI DSS Version 1.2, which was released on October 1, 2008 and is effective as of January 1, 2011.

4.1.1 Applicability

The PCI PA-DSS applies to any payment application that stores, processes, or transmits cardholder data as part of authorization or settlement, unless the application would fall under the merchants PCI DSS validation. PAYware PC, PAYware Transact, PAYware STS, PAYware Mobile, and PCCharge (collectively the PAYware NA Payment Applications) were developed by VeriFone for use by third parties, and therefore are subject to PA-DSS validation. PAYware Connect (also included under PAYware NA Payment Applications) falls under PCI DSS. See the PCI PA-DSS Program Guide to determine if PCI PA-DSS validation is required for any other payment applications used at the merchant location. It is important to note that PA-DSS validated payment applications alone do not guarantee PCI DSS compliance for the merchant. The validated payment application must be implemented in a PCI DSS compliant environment. If your application runs on Windows XP, you are required to turn off Windows XP System Restore Points.

4.1.2 Distribution and Updates

This guide will be provided to VeriFones customers including processors, resellers, ISOs, and integrators along with the PAYware NA payment applications. It is the responsibility of these parties to ensure the information contained in this guide is passed on to their customers (the merchant), in order to illustrate the requirements for complying with PCI DSS. Additional information about PA-DSS and VeriFones PA-DSS Training can be found on our website at www.verifone.com/padss.

4.1.3 What Does PA-DSS Mean to You?

VeriFone submits its PAYware NA payment applications to an annual PA-DSS audit in order to maintain PA-DSS validation. In addition, if any major version change is made in any PAYware NA payment applications, VeriFone submits that application for a full PA-DSS audit again. An example of a major version change would be PCCharge 5.8 moving to PCCharge 5.9. For minor changes, such as PCCharge 5.8.1 to 5.8.2, an attestation form for minor change revisions is required. These minor change revisions are submitted to VeriFones Qualified Security Assessor (QSA), which creates a report to submit to PCI SSC. All validated applications are listed on the PCI SSC web site at https://www.pcisecuritystandards.org/security_standards/vpa/. Always review the full PCI and PA-DSS compliance requirements at the PCISecurityStandards.org web site. Summary PA-DSS requirements material given by VeriFone is not intended to be viewed/used as the full PCI/PA-DSS compliance requirement.

The following table provides opening points to cover in any discussion with merchants on data storage and why a merchant should use PAYware PC, PAYware Transact, PAYware SIM, PAYware Connect, PAYware Mobile, PAYware STS, or PCCharge to assist in maintaining PCI-DSS compliance.

	Data Element	Storage Permitted	Protection Required	PCI DSS Req. 3, 4
Cardholder Data	Primary Account Number	Yes	Yes	Yes
	Cardholder Name [1]	Yes	Yes [1]	No
	Service Code [1]	Yes	Yes [1]	No
	Expiration Date [1]	Yes	Yes [1]	No
Sensitive Authentication Data [2]	Full Magnetic Stripe Data [3]	No	N/A	N/A
	CAV2/CID/CVC2/CVV2	No	N/A	N/A
	PIN/PIN Block	No	N/A	N/A

[1] These data elements must be protected if stored in conjunction with the PAN. This protection should be per PCI DSS requirements for general protection of the cardholder environment. Additionally, other legislation (for example, related to consumer personal data protection, privacy, identity theft, or data security) may require specific protection of this data, or proper disclosure of a company's practices if consumer-related personal data is being collected during the course of business. PCI DSS, however, does not apply if PANs are not stored, processed, or transmitted.

[2] Do not store sensitive authentication data after authorization (even if encrypted).

[3] Full track data from the magnetic stripe, magnetic-stripe image on the chip, or elsewhere.

4.1.4 Third Party Applications

A PAYware NA payment application validation does not extend to any external third party application that has integrated such PAYware NA payment application as the payment engine. The end-to-end transaction process, beginning with entry into the third party application until the response from the payment engine is returned, must meet the same level of compliance. In order to claim the third party application is end-to-end compliant, the application would need to be submitted to a QSA for a full PA-DSS audit.

The use of a PAYware NA payment application also does not exempt a third party integrators application from a PA-DSS audit. The end user and/or P.O.S. developer can integrate and be compliant in the processing portion of a payment transaction. A brief review (given below) of the PA-DSS environmental variables that impact the end user merchant can help the end user merchant obtain and/or maintain PA-DSS compliance. Environmental variables that could prevent passing an audit include without limitation issues involving a secure network connection(s), end user setup location security, users, logging and assigned rights. Remove all testing configurations, samples, and data prior to going into production on your application.

4.1.5 PA-DSS Guidelines

The following PA-DSS Guidelines are being provided by VeriFone as a convenience to its customers. Customers should not rely on these PA-DSS Guidelines, but should instead always refer to the most recent PCI DSS Program Guide published by PCI SSC.

1. **Sensitive Data Storage Guidelines**

   Do not retain full magnetic stripe, card validation code or value (CAV2, CID, CVC2, CVV2), or PIN block data. 1.1 Do not store sensitive authentication data after authorization (even if encrypted): Sensitive authentication data includes

the data as cited in the following Requirements 1.1.1 through 1.1.3. PCI Data Security Standard Requirement 3.2

Note: By prohibiting storage of sensitive authentication data after authorization, the assumption is that the transaction has completed the authorization process and the customer has received the final transaction approval. After authorization has completed, this sensitive authentication data cannot be stored.

1.1.1 After authorization, do not store the full contents of any track from the magnetic stripe (located on the back of a card, contained in a chip, or elsewhere). This data is alternatively called full track, track, track 1, track 2, and magnetic-stripe data.

In the normal course of business, the following data elements from the magnetic stripe may need to be retained:

- The accountholders name,
- Primary account number (PAN),
- Expiration date, and
- Service code
- To minimize risk, store only those data elements needed for business.

Note: See PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms for additional information. PCI Data Security Standard Requirement 3.2.1

1.1.2 After authorization, do not store the card-validation value or code (three-digit or four-digit number printed on the front or back of a payment card) used to verify card-not-present transactions. Note: See PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms for additional information. PCI Data Security Standard Requirement 3.2.2

1.1.3 After authorization, do not store the personal identification number (PIN) or the encrypted PIN block.

Note: See PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms for additional information. PCI Data Security Standard Requirement 3.2.3

1.1.4 Securely delete any magnetic stripe data, card validation values or codes, and PINs or PIN block data stored by previous versions of the payment application, in accordance with industry-accepted standards for secure deletion, as defined, for example by the list of approved products maintained by the National Security Agency, or by other State or National standards or regulations. PCI Data Security Standard Requirement 3.2

Note: This requirement only applies if previous versions of the payment application stored sensitive authentication data.

1.1.5 Securely delete any sensitive authentication data (pre-authorization data) used for debugging or troubleshooting purposes from log files, debugging files, and other data sources received from customers, to ensure that magnetic stripe data, card validation codes or values, and PINs or PIN block data are not stored on software vendor systems. These data sources must be collected in limited amounts and only when necessary to resolve a problem, encrypted while stored, and deleted immediately after use. PCI Data Security Standard Requirement 3.2

2. Protect stored cardholder data

2.1 Software vendor must provide guidance to customers regarding purging of cardholder data after expiration of customer-defined retention period. PCI Data Security Standard Requirement 3.1

2.2 Mask PAN when displayed (the first six and last four digits are the maximum number of digits to be displayed).

Notes:

- This requirement does not apply to those employees and other parties with a legitimate business need to see full PAN;

- This requirement does not supersede stricter requirements in place for displays of cardholder datafor example, for point-of-sale (POS) receipts. PCI Data Security Standard Requirement 3.3

2.3 Render PAN, at a minimum, unreadable anywhere it is stored, (including data on portable digital media, backup media, and in logs) by using any of the following approaches:

- One-way hashes based on strong cryptography with associated key management processes and procedures
- Truncation
- Index tokens and pads (pads must be securely stored)
- Strong cryptography with associated key management processes and procedures. The MINIMUM account information that must be rendered unreadable is the PAN. PCI Data Security Standard Requirement 3.4

The PAN must be rendered unreadable anywhere it is stored, even outside the payment application. Note: Strong cryptography is defined in the PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms.

2.4 If disk encryption is used (rather than file- or column-level database encryption), logical access must be managed independently of native operating system access control mechanisms (for example, by not using local user account databases). Decryption keys must not be tied to user accounts. PCI Data Security Standard Requirement 3.4.2

2.5 Payment application must protect cryptographic keys used for encryption of cardholder data against disclosure and misuse. PCI Data Security Standard Requirement 3.5

2.6 Payment application must implement key management processes and procedures for cryptographic keys used for encryption of cardholder data. PCI Data Security Standard Requirement 3.6

2.7 Securely delete any cryptographic key material or cryptogram stored by previous versions of the payment application, in accordance with industry-accepted standards for secure deletion, as defined, for example the list of approved products maintained by the National Security Agency, or by other State or National standards or regulations. These are cryptographic keys used to encrypt or verify cardholder data. PCI Data Security Standard Requirement 3.6

Note: This requirement only applies if previous versions of the payment application used cryptographic key materials or cryptograms to encrypt cardholder data.

3. Provide secure authentication features

3.1 The payment application must support and enforce unique user IDs and secure authentication for all administrative access and for all access to cardholder data. Secure authentication must be enforced to all accounts, generated or managed by the application by the completion of installation and for subsequent changes after the "out of the box" installation (defined at PCI DSS Requirements 8.1, 8.2, and 8.5.88.5.15) for all administrative access and for all access to cardholder data. PCI Data Security Standard Requirements 8.1, 8.2, and 8.5.88.5.15

Note: These password controls are not intended to apply to employees who only have access to one card number at a time to facilitate a single transaction. These controls are applicable for access by employees with administrative capabilities, for access to servers with cardholder data, and for access controlled by the payment application. This requirement applies to the payment application and all associated tools used to view or access cardholder data. 3.-1.10 If a payment application session has been idle for more than 15 minutes, the application requires the user to re-authenticate. PCI Data Security Standard Requirement 8.5.15.

3.2 Software vendors must provide guidance to customers that all access to PCs, servers, and databases with payment applications must require a unique user ID and secure authentication. PCI Data Security Standard Requirements 8.1 and 8.2

3.3 Render payment application passwords unreadable during transmission and storage, using strong cryptography based on approved standards

Note: Strong cryptography is defined in PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms. PCI Data Security Standard Requirement 8.4

4. Log payment application activity

4.1 At the completion of the installation process, the out of the box default installation of the payment application must log all user access (especially users with administrative privileges), and be able to link all activities to individual users. PCI Data Security Standard Requirement 10.1

4.2 Payment application must implement an automated audit trail to track and monitor access. PCI Data Security Standard Requirements 10.2 and 10.3

5. Develop secure payment applications

5.1 Develop all payment applications in accordance with PCI DSS (for example, secure authentication and logging) and based on industry best practices and incorporate information security throughout the software development life cycle. These processes must include the following: PCI Data Security Standard Requirement 6.3

5.1.1 Live PANS are not used for testing or development. PCI Data Security Standard Requirement 6.4.4.

- Validation of all input (to prevent cross-site scripting, injection flaws, malicious file execution, etc.)
- Validation of proper error handling
- Validation of secure cryptographic storage
- Validation of secure communications
- Validation of proper role-based access control (RBAC)

5.1.2 Separate development/test, and production environments

5.1.3 Removal of test data and accounts before production systems become active development. PCI Data Security Standard Requirement 6.4.4

5.1.4 Review of payment application code prior to release to customers after any significant change, to identify any potential coding vulnerability. Removal of custom payment application accounts, user IDs, and passwords before payment applications are released to customers

Note: This requirement for code reviews applies to all payment application components (both internal and public-facing web applications), as part of the system development life cycle required by PA-DSS Requirement 5.1 and PCI DSS Requirement 6.3. Code reviews can be conducted by knowledgeable internal personnel or third parties.

5.2 Develop all web payment applications (internal and external, and including web administrative access to product) based on secure coding guidelines such as the Open Web Application Security Project Guide. Cover prevention of common coding vulnerabilities in software development processes, to include:

- Injection flaws, with particular emphasis on SQL injection, Cross-site scripting (XSS) OS Command Injection, LDAP and Xpath injection flaws, as well as other injection flaws.
- Buffer Overflow.
- Insecure cryptographic storage.
- Insecure communications.
- Improper error handling.

- All HIGH vulnerabilities as identified in the vulnerability identification process at PA-DSS Requirement 7.1.
- Cross-site scripting (XSS)
- Improper access control such as insecure direct object references, failure to restrict URL access and directory traversal.
- Cross-site request forgery (CSRF)

Note: The vulnerabilities listed in PA-DSS Requirements 5.2.1 through 5.2.9 and in PCI DSS at 6.5.1 through 6.5.9 were current in the OWASP guide when PCI DSS v1.2 / PCI DSS v2.0 (01/01/10) were published. However, if and when the OWASP guide is updated, the current version must be used for these requirements.

5.3 Software vendor must follow change control procedures for all product software configuration changes. PCI Data Security Standard Requirement 6.4. 5.The procedures must include the following:

- Documentation of impact
- Management sign-off by appropriate parties
- Testing functionality to verify the new change(s) does not adversely impact the security of the system. Remove all testing configurations, samples, and data before finalizing the product for production.
- Back-out or product de-installation procedures

5.4 The payment application must not use or require use of unnecessary and insecure services and protocols (for example, NetBIOS, file-sharing, Telnet, unencrypted FTP must be secured via SSH, S-FTP, SSL, IPSec and other technology to implement end to end security). PCI Data Security Standard Requirement 2.2.2

6. Protect wireless transmissions

6.1 For payment applications using wireless technology, the wireless technology must be implemented securely. Payment applications using wireless technology must facilitate use of industry best practices (for example, IEEE 802.11i) to implement strong encryption for authentication and transmission. Controls must be in place to protect the implemented wireless network from unknown wireless access points and clients. This includes testing the end users wireless deployment on a quarterly basis to detect unauthorized access points within the system. Change wireless vendor defaults, including but not limited to default wireless encryption keys, passwords, and SNMP community strings. Maintain a detailed updated hardware list. The end to end wireless implementation must be end to end secure. The use of WEP as a security control was prohibited as of 30 June 2010. PCI Data Security Standard Requirements 1.2.3, 2.1.1, 4.1.1, 6.2, 11.1a-e and 11.4a-c.

7. Test payment applications to address vulnerabilities

7.1 Software vendors must establish a process to identify newly discovered security vulnerabilities (for example, subscribe to alert services freely available on the Internet) and to test their payment applications for vulnerabilities. Any underlying software or systems that are provided with or required by the payment application (for example, web servers, third-party libraries and programs) must be included in this process. Remove all test configurations, samples, and data after testing and before promoting the changes to production. PCI Data Security Standard Requirement 6.2

7.2 Software vendors must establish a process for timely development and deployment of security patches and upgrades, which includes delivery of updates and patches in a secure manner with a known chain-of-trust, and maintenance of the integrity of patch and update code during delivery and deployment.

8. Facilitate secure network implementation

8.1 The payment application must be able to be implemented into a secure network environment. Application must not interfere with use of devices, applications, or configurations required for PCI DSS compliance (for example, payment application cannot interfere with anti-virus protection, firewall configurations, or any other device, application, or configuration required for PCI DSS compliance). PCI Data Security Standard Requirements 1, 3, 4, 5, and 6.

9. Cardholder data must never be stored on a server connected to the Internet

9.1 The payment application must be developed such that the database server and web server are not required to be on the same server, nor is the database server required to be in the DMZ with the web server. PCI Data Security Standard Requirement 1.3.7

10. Facilitate secure remote software updates

10.1 If payment application updates are delivered securely via remote access into customers systems, software vendors must tell customers to turn on remote-access technologies only when needed for downloads from vendor and to turn off immediately after download completes. Alternatively, if delivered via VPN or other high-speed connection, software vendors must advise customers to properly configure a firewall or a personal firewall product to secure authentication using a two factor authentication mechanism. PCI Data Security Standard Requirement 8.3

10.2 If payment application may be accessed remotely, remote access to the payment application must be authenticated using a two factor authentication mechanism. PCI Data Security Standard Requirement 8.3

10.3 Any remote access into the payment application must be done securely. If vendors, resellers/integrators, or customers can access customers payment applications remotely, the remote access must be implemented securely. PCI Data Security Standard Requirements 1, 8.3 and 12.3.9

11. Encrypt sensitive traffic over public networks

11.1 If the payment application sends, or facilitates sending, cardholder data over public networks, the payment application must support use of strong cryptography and security protocols such as SSL/TLS and Internet protocol security (IPSEC) to safeguard sensitive cardholder data during transmission over open, public networks. Examples of open, public networks that are in scope of the PCI DSS are: The Internet Wireless technologies Global System for Mobile Communications (GSM) General Packet Radio Service (GPRS) PCI Data Security Standard Requirement 4.1

11.2 The payment application must never send unencrypted PANs by end-user messaging technologies (for example, e-mail, instant messaging, and chat). PCI Data Security Standard Requirement 4.2

12. Encrypt all non-console administrative access

12.1 Instruct customers to encrypt all non-console administrative access using technologies such as SSH, VPN, or SSL/TLS for web-based management and other non-console administrative access. Telnet or remote login must never be used for administrative access. PCI Data Security Standard Requirement 2.3

13. Maintain instructional documentation and training programs for customers, resellers, and integrators

13.1 Develop, maintain, and disseminate a PA-DSS Implementation Guide(s) for customers, resellers, and integrators that accomplishes the following:

- Addresses all requirements in this document wherever the PA-DSS Implementation Guide is referenced.
- Includes a review at least annually and updates to keep the documentation current with all major and minor software changes as well as with changes to the requirements in this document.

13.2 Develop and implement training and communication programs to ensure payment application resellers and integrators know how to implement the payment application and related systems and networks according to the PA-DSS Implementation Guide and in a PCI DSS-compliant manner.

- Update the training materials on an annual basis and whenever new payment application versions are released.

4.1.6 More Information

VeriFone Systems, Inc. highly recommends that merchants contact the card association(s) or their processing company and find out exactly what they mandate and/or recommend. Doing so may help merchants protect themselves from fines and fraud.

For more information related to security, visit:

- http://www.pcisecuritystandards.org
- http://www.visa.com/cisp
- http://www.sans.org/resources
- http://www.microsoft.com/security/default.asp
- https://sdp.mastercardintl.com/
- http://www.americanexpress.com/merchantspecs

CAPN questions: capninfocenter@aexp.com

4.2 Introduction

This document is designed for developers and integrators interested in integrating their third party iOS applications quickly and securely to the VeriFone PAYware Connect gateway. The purpose of this document is to:

- Define the Application Program Interface (API) between the iOS application and VeriFone MSR devices
- Define the Application Program Interface (API) between the iOS application and VeriFones PAYware Connect gateway

Supported VeriFone MSR devices include:

- PAYware Mobile Sleeve (Gen1 and Gen1.5)
- e210/e255/e315 NOTE: Currently, only the MSR (swipe) is supported for CREDIT transactions on the PAYware Connect gateway. NOTE: e210/e255/e315 must have XPI 5.30 or later installed.
- IDTech UniMag Pro (must be one with VeriFone keys injected)
- VeriFone Secure Audio Reader

4.2.1 Overview

The VeriFone Communication Mobile Framework (VCMF) is a comprehensive solution for third party iOS applications to quickly and securely communicate with the VeriFone PAYware Connect gateway. In addition, VCMF supports approved VeriFone Hardware, such as MSR devices. The integrator does not need to implement External Accessory protocol, or configure input/output stream. A single class (VFIConnect) in VCMF gives the integrator control over a range of supported devices.

VFIConnect was developed to assist iOS integrators in communicating with the PAYware Connect Gateway and, in addition, seamlessly interface with approved VeriFone hardware for capturing card data.

VFIConnect was designed to:

- Handle all network communication and maintain a secondary fallback address if the main address is unavailable.
- Allow easy account registration with a single method call.
- Create unique encrypted string credentials for the integrator to submit with transactions for processing.
- Allow standard transactions to be submitted and responses received with simple key/object dictionary pairs.
- Allow XML passthrough to the gateway.
- Allow XML passthrough from the gateway.
- Convert XML passthrough from gateway to NSDictionary.
- Integrate with VeriFone approved hardware.

4.2.2 Definition of Terms

The following terms and abbreviations are used throughout this document:

Term	Description
API	Application program interface
IBAction	Interface Builder Action. A method that responds to an Interface Builder control action, like a button click. Does not return a value (same as void).
MSR	Magnetic Stripe Reader
PWC	PAYware Connect
POS	POS is a term used to refer to a POS (Point of Sale) Terminal or ECR in this document. The POS connects via serial interface to the MX 800 Series device.
Sled reader	PAYware Mobile sleeve
VCMF	Framework provided by PAYware Connect Mobile SDK
VFIConnect	Class within the VCMF framework
VSP	VeriShield Protect. A security feature offered by VeriFone that allows VeriFone devices to encrypt account information.

4.2.3 Reference Documents

- PAYware Connect Integration Guide

4.3 Installation and Integration Setup

4.3.1 System Requirements

PAYware Connect Mobile for iOS requires the following:

iOS device:

- iPhone, iPod touch, or iPad
- iOS with firmware 4.0 or higher (most current is recommended)
- Apple XCode Developer Tools package

VeriFone MSR device:

- PAYware Mobile Sleeve or VX 600 or VX 600BT or IDTech UniMag Pro (with VeriFone keys injected) or VeriFone Secure Audio Reader

4.3.2 External Accessory Protocol Strings

To use Verifone branded Apple certified external accessories, the project .plist must contain the devices external accessory protocol string. The .plist identifier is "Supported external accessory protocols", which would contain an array of all supported devices.

The correct external accessory protocols strings are as follows:

- PAYware Mobile Sleeve (Gen1 and Gen1.5) - "com.verifone.payware"
- e210- "com.verifone.pmr2.xpi", "com.verifone.pmr2.control", "com.verifone.pmr2.barcode", "com.verifone.pmr2.-zontalk"
- e255 - "com.verifone.PWMRDA"
- e315- "com.verifone.pmr.xpi", "com.verifone.pmr.control", "com.verifone.pmr.barcode", "com.verifone.pmr.-zontalk"

In addition, if the application that uses the external accessory will be distributed in the AppStore and is subject to Apple Review, it must be pre-appoved to use those devices. The application information must be submitted to VeriFone to be included in the MFi (Made for iPod) white list as a supported application for the device(s). Applications built for Enterprise distribution do not require inclusion in the MFi database.

Additional information about External Accessory can be found at apple.com.

4.3.3 Setting up PAYware Connect Mobile Integration

PAYware Connect Mobile Framework (VCMF) includes a Class (VFIConnect) to interface with the PAYware Connect gateway and approved VeriFone MSR devices.

Import VCMF and Other Frameworks

- Open Xcode.
- Create a new Xcode Project.
- Choose and name an application.
- Highlight the Framework directory within your project.

- Right click > Add > Existing Frameworks > Add Other >VCMF.framework (browse to locate).
- Right click > Add > Frameworks. Add the following frameworks:
  - ExternalAccessory.framework
  - CFNetwork.framework
  - AVFoundation.framework
  - AudioToolbox.framework
  - CoreLocation.framework
  - MediaPlayer.framework
  - Security.framework
  - SystemConfiguration.framework
  - libz.1.2.3.dylib or libz.1.2.5.dylib

Modify Your Project Header File

- In your project header file, add the import statement for the VFIConnect class:

```
#import <VCMF/VFIConnect.h>
```

- Make your project class a delegate of the VFIConnect class. For the header interface, add VFIConnectDelegate:

```
@interface ViewController : UIViewController<VFIConnectDelegate>
```

- Set up a VFIConnect Object:
  - For the implementation in .h:

    ```
    VFIConnect *connect;
    ```

  - Set the property in .h:

    ```
    @property(nonatomic, retain) VFIConnect *connect;
    ```

Set up VFIConnect Required Delegate Methods

- Create settors and gettors for connect property by declaring synthesize. Manage memory de-allocation by including connect in dealloc method:
  - Set the synthesize in .m:

    ```
    @synthesize connect;
    ```

  - Set the dealloc in .m:

    ```
    [connect release];
    ```

- Include the required delegate protocol method to receive messages and card data from the framework.

```
- (void) accessoryDataReceived:(AccMessage)message data:(swipeData*)card{
}
```

Initialize VFIConnect Class and Set Delegate

```
- (void)viewDidLoad {
    [super viewDidLoad];
            connect = [[VFIConnect alloc] initWithDelegate:self];
}
```

Note: Alternatively, the VFIConnect class can by initialized by calling initWithLogging. Consult Appendix A VFIConnect Protocol for more information.

Set XCode Compiler Flags

VCMF incorporates some low level C++ code and, as a result, it may be necessary to set a linker flag to avoid XCode build errors. Under your projects Build Setting, add -lstdc++ under Other Linker Flags.

Issue Commands

Note: Consult Appendix A VFIConnect Protocol (see page 33) for a complete API reference. Below is an example of a method call to execute VFIConnect object (using Demo URL).

Example: Register account

```
NSString *credentials =
[connect registerAccount:@"80300010002" userID:@"VFuser" password:@"pass123" gateway:HostAddress_Demo
      customIP:nil];
```

Sample Code

The following example creates an IBAction to request a card swipe. In the delegate method, when the delegate method gets the card data, it will immediately send it off for processing to the PAYware Connect gateway. Once the transaction is approved, the resulting transaction ID will print to the debug console.

Header:

```
#import     <VCMF/VFIConnect.h>

@interface MainViewController : UIViewController<VFIConnectDelegate> {

            VFIConnect *connect;
}

-(IBAction) swipeCard;

@property(nonatomic, retain) VFIConnect *connect;

@end
```

Method:

```
#import "MainViewController.h"

@implementation MainViewController

@synthesize connect;


-(IBAction) swipeCard{
            [connect requestSwipe];
}

- (void) accessoryDataReceived:(AccMessage)message data:(swipeData*)card{
            NSLog(@"Message = %i",message);
            NSLog(@"Card Data =\nMasked Pan: %@\nCard Type: %i\nFirst Name: %@\nLast Name: %@",card.
      maskedPAN, card.cardType, card.firstName, card.
      lastName);
}

- (void)viewDidLoad {
    [super viewDidLoad];
            connect = [[VFIConnect alloc] initWithDelegate:self];
}

- (void)dealloc {
    [super dealloc];
            [connect release];
}

@end
```

4.4 VFIConnect Implementation

4.4.1 Overview

VFIConnect was developed to assist iOS integrators in communicating with the class, making it an object that implements to the PAYware Connect gateway, and in addition, seamlessly interface with approved VeriFone hardware for capturing card data. VFIConnect was designed to:

- Handle all network communication and maintain a secondary fallback address if the main address is unavailable.
- Allow easy account registration with a single method call.
- Create unique encrypted string credentials for the integrator to submit with transactions for processing. VFIConnect will intercept the integrators sensitive credentials during the registration process and convert it into an encrypted string unique to that device/clientID. This encrypted string adds another layer of protection in keeping the merchants sensitive information private. In addition, this encrypted string will provide a single signon for submitting transactions, as it will contain other merchant specific information such as clientID, transactions available for processing, card types accepted, and gateway URL.
- Allow standard transactions to be submitted and responses received with simple key/object dictionary pairs. No need for gateway native XML coding.
- Allow XML passthrough to the gateway. To fully utilize the power of PAYware Connect, some features may require moderate to complex nested XML coding. The integrator has the option to create the XML and submit directly to the gateway. The merchant will be able to use credential substitution (using a single encrypted string for credentials).
- Allow XML passthrough from the gateway. When an integrator implements an XML passthrough call, in most cases, the response will be beyond the scope of VFIConnect to predict/parse. In this case, the integrator has the option of receiving the data as returned from the gateway.
- Convert XML passthrough from gateway to NSDictionary. VFIConnect has an option to parse the XML data and convert it into an NSDictionary. This option may make it easier for the integrator to accept/manage a complex XML data string.
- Integrate with VeriFone hardware. VFIConnect has built in drivers for approved VeriFone card readers. The integrator can leverage the card readers to submit Track Data for preferred rates. VFIConnect will allow the integrator to either submit a transaction with a card swipe request or request a swipe before submitting a transaction to be able to capture/display returned data such as Masked PAN, customer name, and card type. VFIConnect will never expose unmasked Track2 data, only retaining it long enough in memory to submit with the pending transaction or until another event happens that clears it out of memory (e.g., force clear, application exit).

4.4.2 Network Communication

When communicating with the PAYware Connect production gateway, VFIConnect has a primary and secondary IP address. If the primary gateway is unavailable when performing a transaction, the secondary gateway will be checked for availability. If the secondary gateway is unavailable, an error will be returned. VFIConnect can also be configured to use a custom IP address, or it can use VeriFones demo gateway.

The gateway is selected during account registration. Upon successful registration, the gateway is encoded within the string credentials that are returned from the registration method. The gateway to be used can only be configured during account registration.

4.4.3 Account Registration

Account registration is performed by calling one of the following methods:

```
-(NSString*) registerAccount:(NSString*)clientID userID:(NSString*)userID password:(NSString*)password
      serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP;

-(NSString*) registerAccountMultiUser:(NSString*)clientID userID:(NSString*)userID password:(NSString*)
      password serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP;

-(NSString*) registerAccount:(NSString*)clientID userID:(NSString*)userID password:(NSString*)password
      serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP platform:(NSString*)
      type;

-(NSString*) registerAccountMultiUser:(NSString*)clientID userID:(NSString*)userID password:(NSString*)
      password serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP platform:(
      NSString*)type;
```

The integrator will pass clientID, userID, password, `unique serial number`, gateway, and (optional) custom IP address. The gateway is an enumeration and can be one of three values:

```
HostAddress_Demo
HostAddress_Production
HostAddress_Custom
```

If the custom gateway is selected, the customIP parameter must be passed.

This method returns an NSString. If the registration is successful, the NSString will be an encrypted text string to be stored for future use. This will be the merchants encrypted credentials and contains a number of data fields required to process transactions. If the registration is unsuccessful, the NSString returned will be prefixed by VCMF_ERROR: followed by the error description.

4.4.4 Transactions

The main routine to submit transactions is:

```
-(NSDictionary*) submitTransaction:(NSDictionary*)values user:(NSString*)credentials
error:(NSString**)errorText;
```

The values NSDictionary containing XML tags and values needs to be populated, the credentials NSString (acquired from account registration) needs to be passed, and then this method will return the gateway response as an NSDictionary. This method is used for any gateway transaction that is sent as <TRANSACTION> and is a single level deep.

Example:

The PAYware Connect Gateway can accept an XML transaction as follows:

```
<TRANSACTION>
<CLIENT_ID>100010001</CLIENT_ID>
<USER_ID>Manager</USER_ID>
<USER_PW>password</USER_PW>
<FUNCTION_TYPE>PAYMENT</FUNCTION_TYPE>
<COMMAND>SALE</COMMAND>
<PAYMENT_TYPE>CREDIT</PAYMENT_TYPE>
<ACCT_NUM>4005550000000019</ACCT_NUM>
<EXP_MONTH>12</EXP_MONTH>
<EXP_YEAR>11</EXP_YEAR>
<TRANS_AMOUNT>1.00</TRANS_AMOUNT>
<INVOICE>123456</INVOICE>
<MERCHANTKEY>123456789</MERCHANTKEY>
</TRANSACTION>
```

This XML string is suited for use with the submitTransaction method because it is an XML record of type <TRANSACTION>, and the fields do not have nested/child nodes. To send this transaction, you would create an NSDictionary with the following key/value pairs:

values NSDictionary

KEY	Value (NSString)
FUNCTION_TYPE	PAYMENT
COMMAND	SALE
PAYMENT_TYPE	CREDIT
ACCT_NUM	40055500000000019
EXP_MONTH	12
EXP_YEAR	11
TRANS_AMOUNT	1.00
INVOICE	123456

Do not include any fields that have to do with credentials. VFIConnect will extract those credentials from the supplied NSString credentials file and include them with the transaction.

The gateway will reply with a response similar to:

```
<RESPONSE>
            <AUTH_CODE>TAS913</AUTH_CODE>
            <CLIENT_ID>100010001</CLIENT_ID>
            <COMMAND>SALE</COMMAND>
            <INVOICE>123456</INVOICE>
            <PAYMENT_MEDIA>MC</PAYMENT_MEDIA>
            <PAYMENT_TYPE>CREDIT</PAYMENT_TYPE>
            <REFERENCE>710717501187</REFERENCE>
            <RESPONSE_TEXT>EXACT MATCH </RESPONSE_TEXT>
            <RESULT>CAPTURED</RESULT>
            <RESULT_CODE>4</RESULT_CODE>
            <TERMINATION_STATUS>SUCCESS</TERMINATION_STATUS>
            <TRANS_AMOUNT>1.00</TRANS_AMOUNT>
            <TRANS_DATE>2011.04.17</TRANS_DATE>
            <TRANS_SEQ_NUM>11</TRANS_SEQ_NUM>
            <TRANS_TIME>14:16:42</TRANS_TIME>
            <CTROUTD>50370</CTROUTD>
</RESPONSE>
```

VFIConnect will parse the response and return an NSDictionary with the following values:

KEY	Value (NSString)
AUTH_CODE	TAS913
CLIENT_ID	100010001
COMMAND	SALE
INVOICE	123456
PAYMENT_MEDIA	MC
PAYMENT_TYPE	CREDIT
REFERENCE	710717501187
RESPONSE_TEXT	EXACT MATCH
RESULT	CAPTURED
RESULT_CODE	4
TERMINATION_STATUS	SUCCESS
TRANS_AMOUNT	1.00
TRANS_DATE	2011.04.17
TRANS_SEQ_NUM	11
TRANS_TIME	14:16:42
CTROUTD	50370

4.4.5 Direct XML Implementation

XML Passthrough

To allow full access to PAYware Connect gateway functions, VFIConnect will allow sending direct XML. VFIConnect will not modify the XML string other than to insert the credentials. This method is used to submit XML string that is not a single level between <TRANSACTION> tags and a response that is not a single level between <RESPONSE> tags:

```
-(NSString*) submitXML:(NSString*)xml user:(NSString*)credentials
error:(NSString**)errorText;
```

Example:

You want to perform an ADDON_NEW_CUSTOMER. The XML string you would create is:

```
@verbatim
<AdminRequest>
  <AdminHeader>
    <RequestType>IPCRB_ADMIN</RequestType>
    <FunctionType>ADDON</FunctionType>
    <CommandType>ADDON_NEW_CUSTOMER</CommandType>
  </AdminHeader>
  <AdminBody>
    <HEADER>
      <Client_ID></Client_ID>
</HEADER>
    <DETAIL>
<RECORD>
        <CUSTOMER Merchant_Customer_ID="CST0">
          <Company_Name>Verifone</Company_Name>
          <Salutation>Mr</Salutation>
          <First_Name>Manager</First_Name>
          <Last_Name>Smith</Last_Name>
          <Middle_Initial>D</Middle_Initial>
          <Address_1>8001 Happy Street</Address_1>
          <Address_2>Suite 300</Address_2>
          <City>Any City</City>
          <State>Any State</State>
          <Zip>12345</Zip>
          <Pri_Contact>Manager</Pri_Contact>
          <Pri_Phone>555-555-5555</Pri_Phone>
          <Pri_Email>email@verifone.com</Pri_Email>
          <Alt_Contact>John Smith</Alt_Contact>
          <Alt_Phone>555-444-4444</Alt_Phone>
          <Alt_Email>email2@verifone.com</Alt_Email>
        </CUSTOMER>
      </RECORD>
    </DETAIL>
  </AdminBody>
</AdminRequest>
@endverbatim
```

Leave a placeholder where the credentials should appear by including <CLIENT_ID></CLIENT_ID>. This will be replaced by tags/values for Client ID, Device Key, Serial Number, Card Reader Serial Number, and Device Type, extracted from the supplied credentials text file. You must include the CLIENT_ID tag, otherwise VFIConnect will not know where in the XML string to insert the values. (Note that you do not need to include the tags in the startTransaction function because VFIConnect knows there is only one level in that method.)

This method will return an XML response from the gateway. The user can parse this response or read the property NSDictionary* dictResponse and receive the response as a dictionary. The XML response can also be read by the property.

XML Response

VFIConnect has a property that stores the most recent XML response from the gateway:

```
@property (readonly) NSString* xmlResponse
```

This response is recorded for all transactions initiated by either submitTransaction or submitXML.

4.4.6 NSDictionary Response

VFIConnect has a property that stores the most recent XML response from the gateway as a dictionary:

```
@property (readonly) NSDictionary* dictResponse
```

This response is parsed from the xmlResponse. It contains an NSDictionary with either NSDictionary for child nodes, or a single key name text with a NSValue representing the NSString value on that node.

Example:

```
<Node1>
   <Node2a>
      <myTag>myValue</myTag>
   </Node2a>
   <myTag2>myValue2</myTag2>
<Node1>
```

Main_NSDictionary_Returned (2 objects, dict and string)

```
Key:Node2a   Value:NSDictionary (1 object)
                      Key:text      Value:myValue (nsstring)
Key:text      Value:myValue2 (nsstring)
```

Returned in the example is an NSDictionary with two objects: the first object is another NSDictionary,and the second object is anNSString value with key named text.

The first object (NSDictionary) contains a single entry with a key named text and a NSString value of myValue.

4.4.7 Hardware Connectivity

All the communication from the attached accessories will pass a message to the delegate protocol:

```
- (void) accessoryDataReceived:(AccMessage)message data:(swipeData*)card{
}
```

For example, if a sled reader is disconnected, it will pass AccMessage_SledDisconnected. If there is card swipe data available, it will pass AccMessage_TrackDataReceived (and the swipeData class will be populated). You can use the accessoryDataReceived along with the AccMessages to monitor accessory connects/disconnects, audio reader power up, processing, and ready states, and accept card data.

Code	AccMessage	Description
0	AccMessage_TrackDataReceived	Track Data Received
1	AccMessage_ReaderTimeout	Reader Timeout
2	AccMessage_SledDisconnected	Sled Reader Disconnected
3	AccMessage_SledConnected	Sled Reader Connected
4	AccMessage_PinpadDisconnected	PIN pad Disconnected
5	AccMessage_PinpadConnected	PIN pad Connected
6	AccMessage_AudioMSR-Disconnected	Audio Reader Disconnected
7	AccMessage_AudioMSRConnected	Audio Reader Connected
8	AccMessage_AudioMSRProcessing	Audio Reader Processing
9	AccMessage_AudioMSRWaiting-ForSwipe	Audio Reader Please Swipe Card
10	AccMessage_AudioMSRTimeout	Audio Reader Timeout
11	AccMessage_AudioMSRPowering-Up	Audio Reader Powering Up

12	AccMessage_UnableToReadTrack	Unable to Read Track
13	AccMessage_VSPTrackCaptured	VSP Track Captured
14	AccMessage_CardNotAccepted	Card Not Accepted
15	AccMessage_ReaderNotReady	Reader Not Ready
16	AccMessage_TrackCapturedFor-Transmission	Track Data Captured Completing Transaction
17	AccMessage_SwipeTransmit-Timeout	Timeout when executing swipe during transmission
18	AccMessage_SwipeCancelled	Cancel Swipe Completed
19	AccMessage_KeepAliveDetected	Audio Reader Keep Alive Detected
20	AccMessage_FinishedRecording	Audio Reader Finished Recording

4.4.8 Track Data

Capturing Track Data

Initiate track data capture by calling the requestSwipe method or by submitting a transaction requiring track data. In the latter case, the captured track data is immediately transmitted to the gateway. To request a swipe, execute:

```
-(void) requestSwipe;
```

If track data is captured, the delegate will receive a message of type AccMessage_TrackDataReceived and data of type swipeData:

```
- (void) accessoryDataReceived:(AccMessage)message data:(swipeData*)card{
}
```

The swipeData class is:

```
NSString*   maskedPAN;
CardType    cardType;
NSString*   firstName;
NSString*   lastName;
```

Integrators can use the data within swipeData class to populate fields in their apps. The full track data is not exposed to the integrator and is retained in VFIConnect memory until it is used to transmit to the gateway, the application exists, or the integrator executes the method:

```
-(void)clearTrackData
```

4.4.9 Sending Track Data

There are two methods to submit track data to the gateway: One is to pass a token identifier with the transaction and have VFIConnect replace it with previously capture track data, and the second is to request the track to be captured during transaction submission.

Refer to the previous example for submitTransaction (page 24):

```
<TRANSACTION>
<CLIENT_ID>100010001</CLIENT_ID>
<USER_ID>Manager</USER_ID>
<USER_PW>password</USER_PW>
<FUNCTION_TYPE>PAYMENT</FUNCTION_TYPE>
<COMMAND>SALE</COMMAND>
<PAYMENT_TYPE>CREDIT</PAYMENT_TYPE>
<ACCT_NUM>4005550000000019</ACCT_NUM>
<EXP_MONTH>12</EXP_MONTH>
<EXP_YEAR>07</EXP_YEAR>
<TRANS_AMOUNT>1.00</TRANS_AMOUNT>
<INVOICE>123456</INVOICE>
<MERCHANTKEY>123456789</MERCHANTKEY>
</TRANSACTION>
```

KEY	Value (NSString)
FUNCTION_TYPE	PAYMENT
COMMAND	SALE
PAYMENT_TYPE	CREDIT
ACCT_NUM	40055500000000019
EXP_MONTH	12
EXP_YEAR	07
TRANS_AMOUNT	1.00
INVOICE	123456

Instead of submitting tags ACCT_NUM, EXP_MONTH, EXP_YEAR, you can submit TRACK_DATA.

```
<TRANSACTION>
    <CLIENT_ID>100010001</CLIENT_ID>
    <USER_ID>Manager</USER_ID>
    <USER_PW>password</USER_PW>
    <FUNCTION_TYPE>PAYMENT</FUNCTION_TYPE>
    <COMMAND>SALE</COMMAND>
    <PAYMENT_TYPE>CREDIT</PAYMENT_TYPE>
    <TRACK_DATA>raw track data strip</TRACK_DATA>
    <TRANS_AMOUNT>1.00</TRANS_AMOUNT>
    <INVOICE>123456</INVOICE>
    <MERCHANTKEY>123456789</MERCHANTKEY>
</TRANSACTION>
```

The merchant does not have access to the raw track data information. If the merchant previously swiped a card in which VFIConnect returned Masked PAN data, the merchant can use that as a token identifier. VFIConnect will replace it with the full raw track data on submission:

KEY Value (NSString) FUNCTION_TYPE PAYMENT COMMAND SALE PAYMENT_TYPE CREDIT TRACK_DATA 400555********0019 TRANS_AMOUNT 1.00 INVOICE 123456

If VFIConnect receives a transaction request with the tag TRACK_DATA, but no value is provided, VFIConnect will initiate the card swipe request, wait for a card swipe, and then immediately send the transaction for processing. If it is a bad read or a timeout, then the method will return an error code. The following values will execute a card swipe request when submitted:

KEY	Value (NSString)
FUNCTION_TYPE	PAYMENT
COMMAND	SALE
PAYMENT_TYPE	CREDIT
TRACK_DATA	(blank string @)
TRANS_AMOUNT	1.00
INVOICE	123456

When submitting XML transactions, use the masked PAN data token for previously captured track data as follows:

```
<TRACK_DATA>400555******0019</TRACK_DATA>
```

Or to initiate a card swipe on submission:

```
<TRACK_DATA></TRACK_DATA>
```

4.5 Unique Serial Number Requirement

Effective May 1st, 2013, Apple will not approve any applications submitted to the AppStore (either as a new app or an update) that uses the following deprecated method for retrieving the iOS device unique identifier:

```
[[UIDevice currentDevice] uniqueIdentifier]
```

The PAYware Connect gateway requires a unique device identifier, also referred to as a device serial number, to be passed before an account can be registered. Up to and including VCMF version 1.0.3.188, the framework automatically provided this value using the now deprecated method uniqueIdentifer. To pass Apple's approval guidelines, beginning with VCMF 1.0.3.189, the call to uniqueIdentifier has been removed from VCMF, leaving it up to the integrator to provide a similiar unique identifier when registering an account using any of the following methods:

```
-(NSString*) registerAccount:(NSString*)clientID userID:(NSString*)userID password:(NSString*)password
      serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP;

-(NSString*) registerAccountMultiUser:(NSString*)clientID userID:(NSString*)userID password:(NSString*)
      password serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP;

-(NSString*) registerAccount:(NSString*)clientID userID:(NSString*)userID password:(NSString*)password
      serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP platform:(NSString*)
      type;

-(NSString*) registerAccountMultiUser:(NSString*)clientID userID:(NSString*)userID password:(NSString*)
      password serialNumber:(NSString*)serialNumber gateway:(HostAddress)addr customIP:(NSString*)customIP platform:(
      NSString*)type;
```

NOTE:

The prior registerAccount methods have been REMOVED from VCMF. When updating a project using VMCF prior to 1.0.3.189 to a version 1.0.3.189 or later, if these methods were incorporated, Xcode will warn of a signature mismatch. The integrators code must be modified to pass the additional serialNumber field.

4.5.1 Generating an unique ID - iOS6

Apple has provided an alternate method for retrieving a unique ID in iOS 6

```
[[[UIDevice currentDevice] identifierForVendor] UUIDString]
```

NOTE:

Since this method is only available to a device running iOS6 or later, a device running an earlier iOS attempting to call this method will crash the application.

4.5.2 Generating an unique ID - all iOS

A general method to generate a unique UDID for all iOS versions is

```
- (NSString *)createNewUUID {

    CFUUIDRef theUUID = CFUUIDCreate(NULL);
    CFStringRef string = CFUUIDCreateString(NULL, theUUID);
    CFRelease(theUUID);
    return [(NSString *)string autorelease];
}
```

NOTE:

This method generates a random UDID every time it is called. To effectively implement this for PAYware Connect usage, it is recommended after the initial UDID is created for a device, it is stored in the application's keychain. This will persist the UDID between updates and application re-installations.

Flow:

- Check keychain for existing UDID
- If exists, use that UDID for all account registration methods

- If it doesn't exist, create new UDID
- Store UDID in application keychain
- Loop back to start - Check keychain for existing UDID and then use for all account registration methods

Chapter 5

VMF API Reference Guide

The VeriFone Mobile Framework (VMF) is an iPhone Framework that will be provided by VeriFone as the main interface between iPhone applications wishing to utilize the e210/e255/e315 payment device for facilitating the capture of customer information (track data, PIN, EMV chip, etc.) necessary to process payments with the iPhone applications designated host processor. The VM- Framework will provide access to the individual XPI and barcode command set as well as provide for more convenient flow function commands in order to simplify the burden on the application to know the complete series of commands necessary to implement a fully-featured payment flow with the customer.

The purpose of this document is to describe the requirements of the VM-Framework as well as the interface definitions and requirements needed for any iPhone applications wishing to deploy with the payment framework.

- Overall Solution Design
- Scope Statement
- Application Software Components
- Implementation Requirements
- Implementation Considerations
- Framework Interface
- EMV Contact-based Initialization
- EMV Contactless Initialization
- Financial Transaction Commands
- Core Implementation using VMF
- Best Practices
- Secure Message Table
- PINPad Parameters
- EMV Data Formats

- General Message Table
- Multi-Language Support
- Country Specific Implementation Considerations
- Troubleshooting

5.1 Overall Solution Design

The Figure 1 diagram below outlines the main components of the overall design solution that will be utilized for the iPhone/iPod Touch payment architecture. Processing elements depicted in the iPhone/iPod Touch layer (above the VM-Framework) are simply examples of what an application may have deployed and are at the discretion of the Apple developer. The VM-Framework is the required software component that will be made available to iPhone/iPod Touch integrators to provide full featured payment solutions within their applications. The VM-Framework will be the interface point to the attached e210/e255/e315 device that will provide the magnetic swipe, EMV chip, contactless (MSD and EMV), and barcode readers as well as a user interface and keyboard to drive message displays and user input gathering in order to gather the information necessary to complete a financial transaction.

It should be noted that prompting will need to occur at both the merchant and the consumer level. The merchant will answer prompts via the iPhone or iPod Touch application and the consumer will respond to prompts via the e210/e255/e315 payment terminal keypad. In general, the delineation of the prompts is as follows:

a. iPhone/iPod Touch App Prompts (the responsibility for providing these prompts and data elements rests with the developer integrating the VeriFone Mobile Framework into their application)

- Transaction/Sale Amount
- Tax Amount
- Surcharge Amount
- Transaction Type
- Zip (Postal) Code
- CVV Data

b. e210/e255/e315 Consumer Prompts

- Tip Amount
- Cash Back Amount
- Surcharge Amount Confirmation (Yes/No)
- PIN Entry

Each major component in the software solution will be described in this section with the exception of the VxOS-11 components which are not in scope for this document.

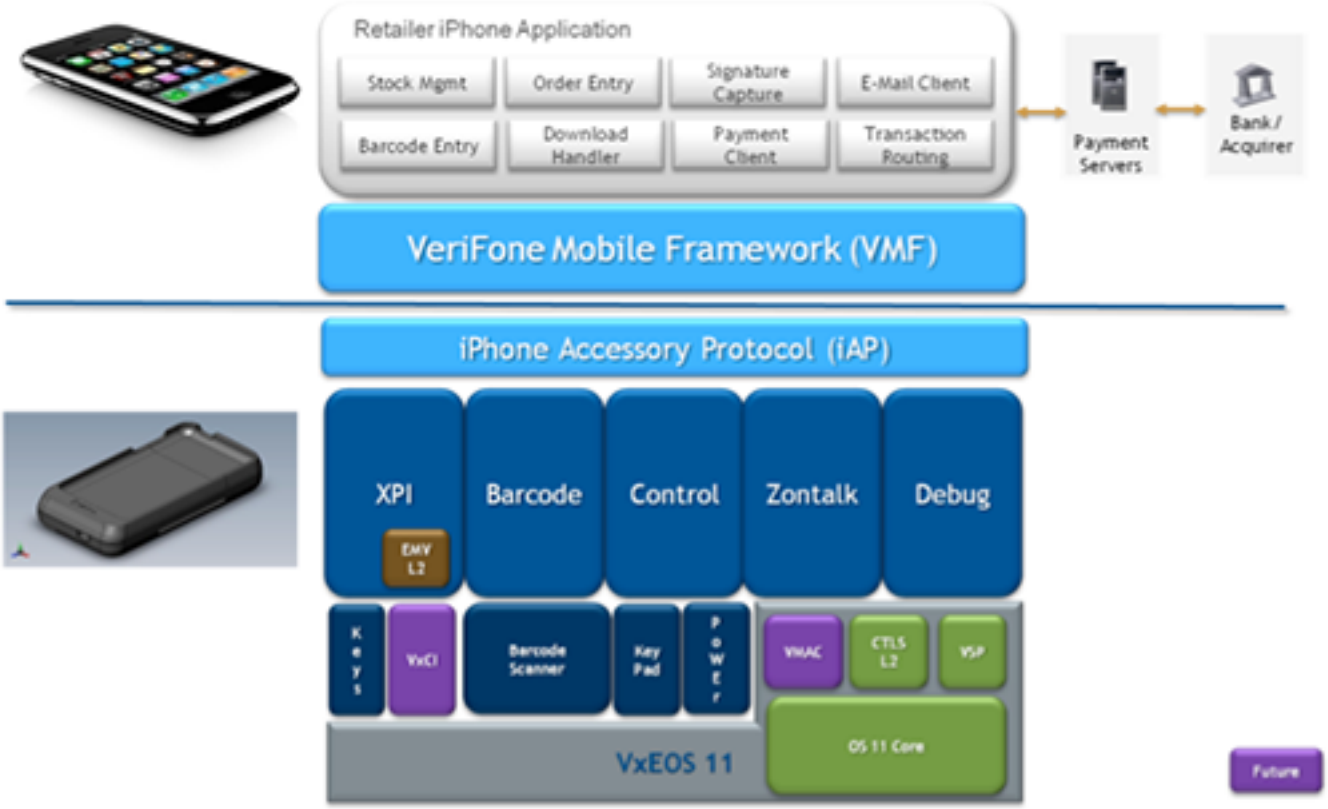


Figure 1: Solution Design Diagram

5.2 Scope Statement

The scope of the initial rollout and deployment of the e210/e255/e315 framework and accompanying XPI will include the following:

- XPI (Handling EMV, Credit, and Debit card processing)
- Barcode Scanner Features
- Download Support through the Mobile Framework using the Zontalk channel and initiated thru XPI

The following features and functionality will be excluded in the initial rollout and deployment and will be added in a later phase:

- MSD and EMV Contactless (VxCI)
- VMAC
- EMA (Estate Management)
- VSP Terminal Generated Keys
  - VSP handled exclusively through command cards for all key initiation and key rotation. e210/e255/e315 updates will be handled through MPF hooks allowing for the push of an application to the pin pad.

5.3 Application Software Components

This section outlines the software applications and libraries directly related to the overall solution design that will be resident on the either the iPhone/iPod Touch device and/or the e210/e255/e315 terminal device and provide the functionality to facilitate the gathering of required information to complete a financial transaction (see Figure 2). These software components will not be responsible for communicating directly to the host processor, but will provide all of the details needed to generate a host processor request transaction, and, in the case of EMV, provide EMV-certified processing with the chip card in accordance with MasterCard TIP, ADVT (Visa), and Interac requirements.

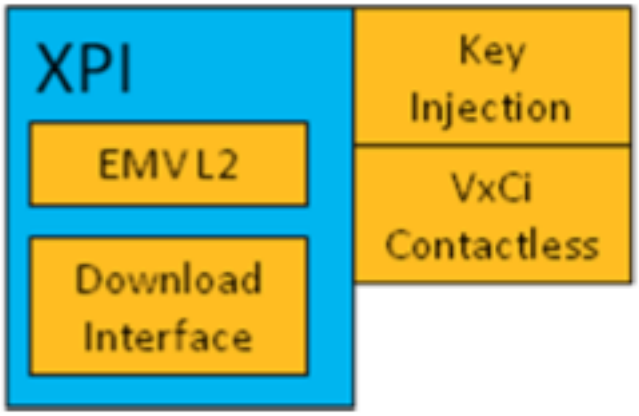
VeriFone Mobile (VMF) Framework

The VM-Framework provides the interface between an iPhone/iPod Touch application and the capabilities of the e210/e255/e315 device to facilitate payment processing. This framework will be an objective-C based library constructed as per Apples Framework requirements and will be made available for inclusion in the iPhone/iPod Touch application build.

iAP Protocol Translator (iPay)

The iAP Protocol Translator, or iPay, will reside on the VeriFone e210/e255/e315 device and will be a pluggable operating system module that can be independently managed and updated.

XPI



XPI is a Verix V application that will reside on the e210/e255/e315 terminal and be the main application responsible for driving the terminal functions necessary to service client requests passed thru the VM-Framework and iPay. XPI handles

obtaining card information for credit, debit, EMV chip, and contactless cards as well as handling message displays on the e210/e255/e315 screen and prompting for additional information that may be relevant to the financial transaction, such as tip, cash back, and surcharge acceptance.

VxCI (Verix V Contactless Interface)

The VxCI application resides on the e210/e255/e315 terminal device and is a companion application to XPI that provides the interface between XPI and the contactless module incorporated into the e210/e255/e315 terminal. The VxCI application supports both MSD and EMV contactless cards depending upon the configuration settings.

Key Injection

Key injection is a process by which keys are loaded into the e210/e255/e315 terminal device to facilitate the secure entry of PINs for debit card transactions. This process is typically done at deployment by the host processor as the keys loaded are unique to them.

Key Loader

In the case of EMV, additional MAC keys are loaded at deployment that configures the terminal to communicate with the designated host for processing EMV transactions. This is typically a temporarily loaded application that is subsequently removed after the MAC keys have been established and is noted herein to identify this component as a consideration, especially since some processors require a unique key loading protocol.

Download Interface

XPI will also provide the mechanism to put the e210/e255/e315 terminal in download mode to allow for transfer of application files for updating applications residing on the terminal device.

EMV Vx 5.0

The EMV Vx 5.0.0 kernel library is a VeriFone product that is EMVCO certified and will be linked with XPI in order to provide the EMV chip card processing to fulfill the requirements for EMV certification.

Barcode Access

Access to the barcode reader will be available through the VM-Framework to the iPay module. The framework will simply provide the contents of the barcode read to the client application for its own processing.

5.4 Implementation Requirements

As described in the previous section, there are a number of components required in order to implement a solution with the VeriFone e210/e255/e315 PINPad accessory and the supporting VeriFone Mobile Framework. Those components and client application requirements and the placement of these components will be discussed in this section of the document.

e210/e255/e315 PINPad Software Requirements

The components that are required to be present on the VeriFone e210/e255/e315 PINPad in order to complement the VM-Framework are as follows:

- XPI (external PINPad Interface)
- VxCI (Contactless Interface)
- iPay (Protocol Translator Interface)

These components are described in more detail in the previous section of this document.

iPhone/iPod Touch Software Requirements

To facilitate access to the application software and devices on the VeriFone e210/e255/e315 PINPad, the client application must include the VM-Framework Library and use the messages provided to interact with the PINPad to perform requests to gather financial data from the cardholder for the purposes of issuing a request to the designated host for authorization. It is up to the client application to handle the host interaction and transaction formatting. The VM-Framework simply provides the necessary data to facilitate the requests with the host. In the case of EMV, the client application must also accommodate the process to initialize relevant EMV data and configuration settings as determined by the host processor.

Download Support

The VM-Framework provides a download message request that enables the client application to transfer the contents of an application or data file to the PINPad device. XPI will put the PINPad into download mode and transfer the contents of the designated file. This process is required in order to update the operating system and software on the PINPad. The framework library does not provide for the transfer of application files to the iPhone for download to the PINPad. This is left to the client application to gather any updated applications or data files and initiate the transfer of that data to the PINPad device through the provided framework message.

Keys

In order to perform debit and EMV transactions, MAC and debit keys are required to be loaded onto the PINPad device. These keys are used to encrypt the PIN and MAC data to be sent to the host for authorization. There are two main methods for loading keys on the PINPad. Both of which will occur without the iPhone or iPod Touch device being connected to the PINPad.

KeyLoader

The KeyLoader is a VeriFone software product that interacts with processors PC key loading software for injecting the PINPad with MAC and Interac keys required for processing financial transactions.

Debit Key Injection

Debit Key injection is supported by the PINPads operating system and will support specialized key injection software to configure debit keys on the PINPad for PIN encryption.

VeriShield Protect

VeriShield Protect provides encryption protection for track data read from the magstripe reader on the PINPad device. In order to fully implement a VSP-enabled solution, the host processor is required to implement a Decryption Appliance (DA) that will serve to decrypt the transferred track data prior to authorization processing. An additional process to periodically update the transport keys on the PINPad is also required.

5.5 Implementation Considerations

There are a number of additional considerations that must be taken into account when configuring the framework and the e210/e255/e315 software components. The following section discusses those considerations and any recommendations to address them.

Transaction Configuration

The VeriFone Mobile Framework will implement a configuration file that will determine which consumer prompts will be enabled or disabled when operating a convenience function (e.g. Purchase). This configuration will also take into consideration the type of transaction that is being executed. For example, for a refund, prompting for cash back or a tip amount would not be required. The configuration settings will determine which XPI prompts that the Mobile Framework will send to XPI on a transaction by transaction basis. This configuration file can be updated by the

integrator before compiling the framework with their iPhone application or the integrator can build a configuration utility within their iPhone mobile application to update the configuration file. A method call or calls will be provided by the framework to perform these settings operations.

BIN (Card Range) Configuration

The type of processing allowed for a given card is determined in large part based on the definition of the range with which the card used by the consumer falls within. These ranges identify whether the card can be used for debit, credit, EMV, contactless, etc. The rules by which these operations are performed is traditionally determined by the host processor in conjunction with the issuer. To facilitate this, the VeriFone Mobile Framework will maintain a BIN range file. The BIN ranges and settings can be updated as needed and deployed with the iPhone application. A default BIN file will be provided that can be dynamically updated and override the defaults when populated. For those hosts that support an initialization process, the BIN file will be updated during the initialization of the device with the host. The framework will provide method(s) to facilitate these updates.

EMV Initialization

XPI requires EMV configuration settings to be established prior to the execution of any EMV transaction (both contact-based and contactless). These configuration settings are determined by the host processor and are set up during the initialization of the device. The settings can also be updated as needed based on the rules/needs of the host processor. This initialization process can be a high volume series of transactions in terms of duration and packet size and should be taken into consideration in the implementation. It is advisable for iPhone or iPod Touch application to gather all of the EMV configuration settings and validate them prior to using the framework to pass the configuration to XPI. Due to the criticality of these configuration settings, it is also advisable to ensure that the iPhone/iPod Touch device is on external power and is disabled from going to sleep/hibernation mode during this process. More details on the EMV initialization can be found in the Initialization Commands section of this document.

Localization

The VeriFone Mobile Framework provides for a language indicator to be passed with any calls that result in pre-defined messages displayed on the e210/e255/e315 PINPad. However, the language set supported by XPI is pre-defined and must be properly configured on the PINPad during installation or a software update. For any language not already supported by XPI, a new language index will need to be defined and a corresponding language file deployed to the PINPad containing the appropriate language translations for each pre-defined prompt appearing in the message file. It is ultimately the responsibility of the iPhone developer to provide localization of the text, prompts, and messages appearing within their iPhone application.

Software Updates (Downloads)

The e210/e255/e315 PINPad device will support downloads of the operating system, software applications, and configuration data. The VeriFone Mobile Framework provides a method call that allows the iPhone application to provide a type designation (file or application) and file stream to use in transferring the data from the iPhone/iPod Touch to the e210/e255/e315 PINPad. The framework will utilize a VeriFone standard download protocol (ZONTAL-K) to execute the download which will then update the device and reboot. It is recommended that the binary files for any file to be deployed to the e210/e255/e315 PINPad be embedded into the iPhone/iPod Touch application which can then be updated from the iPhone to the e210/e255/e315 device locally. This will ensure that the updates have been tested and certified with the released iPhone application. It is, however, the discretion of the iPhone developer to utilize any method desired to obtain the software updates. The framework only requires the ability to be able to utilize a provided file stream to transfer the data to the e210/e255/e315 device. Updates to the e210/e255/e315 PINPad may include at a minimum:

- Operating System
- iAP (iPay) O/S Pluggable Module
- XPI
- KeyLoader

- VxCI
- Barcode Reader Application & Firmware
- Contactless Reader Firmware
- VSP Firmware
- Configuration Parameters (config.sys)

VSP

VeriShield Protect configuration will follow the same process as currently implemented for the GEN1 device. Devices requiring VSP configuration will be activated at the time of deployment. The complete requirements for VSP configuration will be documented and reviewed for security considerations. The following requirements will require verification and adopted as part of the e210/e255/e315 solution:

- e210/e255/e315 devices for non-PAYWare Mobile use can be shipped from the deployment center with VSP disabled as required by the integrator.
- e210/e255/e315 devices for PAYWare Mobile US will be shipped with VSP activated between the PAYWare Connect Gateway and the Semtek Decryption Appliance (DA).
- e210/e255/e315 devices will not allow VSP to be deactivated (turned off) in the field. Not adhering to this policy introduces the risk of having rogue applications/users disabling the security features and using these devices for skimming.
- When VSP is active, manually entered and contactless card data will require encryption by XPI.
- VSP Key Management for initial deployment of e210/e255/e315 devices will be done using command cards (same as GEN1 devices).

5.6 Framework Interface

EMV is a secure method of protecting against fraudulent activity through authenticating and validating card holder present transactions. CHIP transactions use cryptography to validate the card has been certified by the Issuer and Scheme. The Scheme i.e. (visa) provides the supported financial processor with the public keys. Prior to processing a CHIP transaction the terminal must be first configured to support EMV. The financial processor will provide parameters either through a file or real time messages that provide instructions for the terminal and ECR. The processor(s) parameters contain pertinent information such as the card brands supported, the different applications supported, the public keys and the processing rules for each card. Please note each processor may have different rules and protocols which must be adhered to.

5.6.1 XPI Tables

EST

The EST table contains the card brands and applications which will be supported by the processor. Each card issuer may support multiple applications for chip transactions. There will be two identifiers found in the EST table, an Registered Application Provider Identifier (RID) which is used to determine the card brand and an Application Identifier (AID) which indicates the different applications each card brand supports i.e. (DEBIT, CREDIT).

- RID
  - A000000003 Visa
  - A000000004 MasterCard
  - A000000025 American Express
  - A000000277 Interac

- AID
  - A0000000031010 Visa Credit/Debit
  - A0000000031020 Visa Delta
  - A0000000041010 MasterCard Credit
  - A0000000043060 MCard Maestro Debit

MVT

The MVT table contains the parameters the processor has setup for each of the applications supported. The MVT table parameters define how the terminal and card react under certain conditions.

EEST

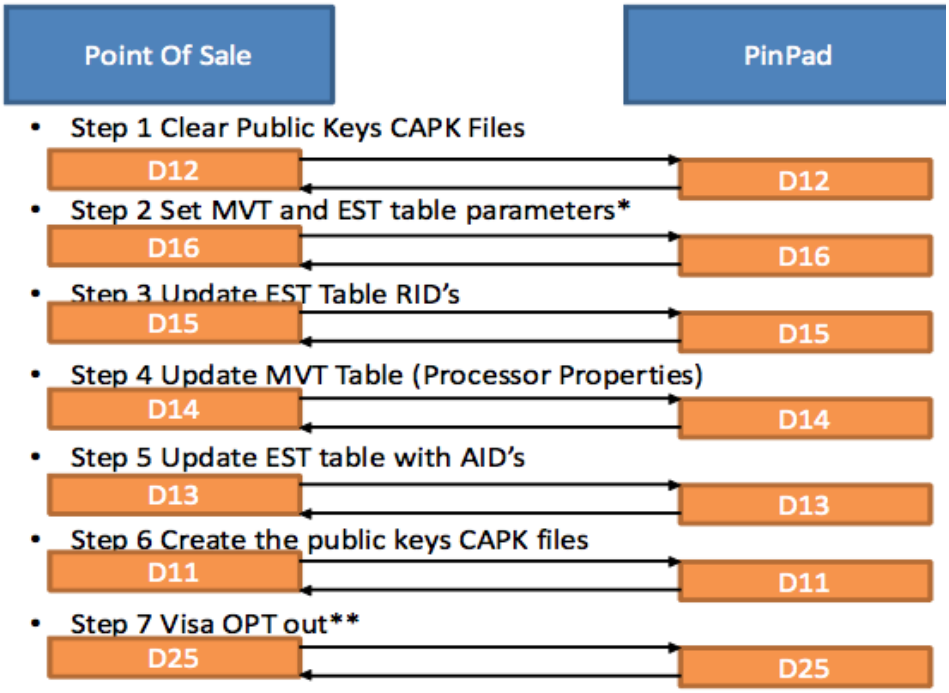
The Extended EST table is required for contactless transactions. The EEST table needs to be populated with the AIDs for all applications supported contactless transactions.

EMVT

The Extended MVT table is required for contactless transactions. The EMVT table provides the rules engine for how to handle EMV contactless transactions.

5.7 EMV Contact-based Initialization

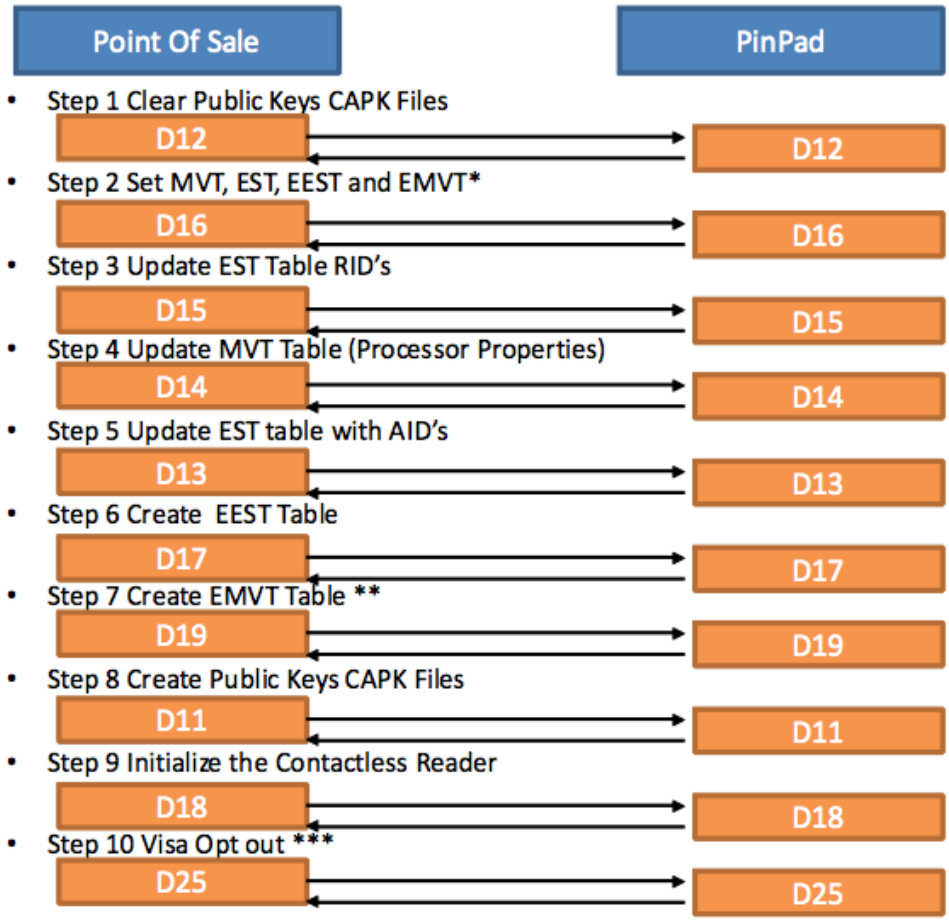
Below are the message commands between the integrators application and XPI in order to support configurations supporting contact-based chip transactions. The VeriFone Mobile Framework provides methods that abstract the calling application from implementing the XPI protocol and enable the application to pass the required EMV configuration data to XPI to establish the rules for EMV Contact-based transaction processing. These steps need to be completed prior to an EMV transaction taking place. The actual XPI command code is listed under the Notes section of each framework command.



- There will be multiple D16, D15 and D13 commands sent for each card type supported
- To support Visa opt out <modifyCandListFlag> needs to be set to one (1) in the D14 command.

5.8 EMV Contactless Initialization

Below are the message commands between the integrators application and XPI in order to support configurations supporting contact-based and contactless chip transactions. The VeriFone Mobile Framework provides methods that abstract the calling application from implementing the XPI protocol and enable the application to pass the required EMV configuration data to XPI to establish the rules for EMV Contact and Contactless transaction processing. These steps need to be completed prior to an EMV transaction taking place. The actual XPI command code is listed under the Notes section of each framework command.



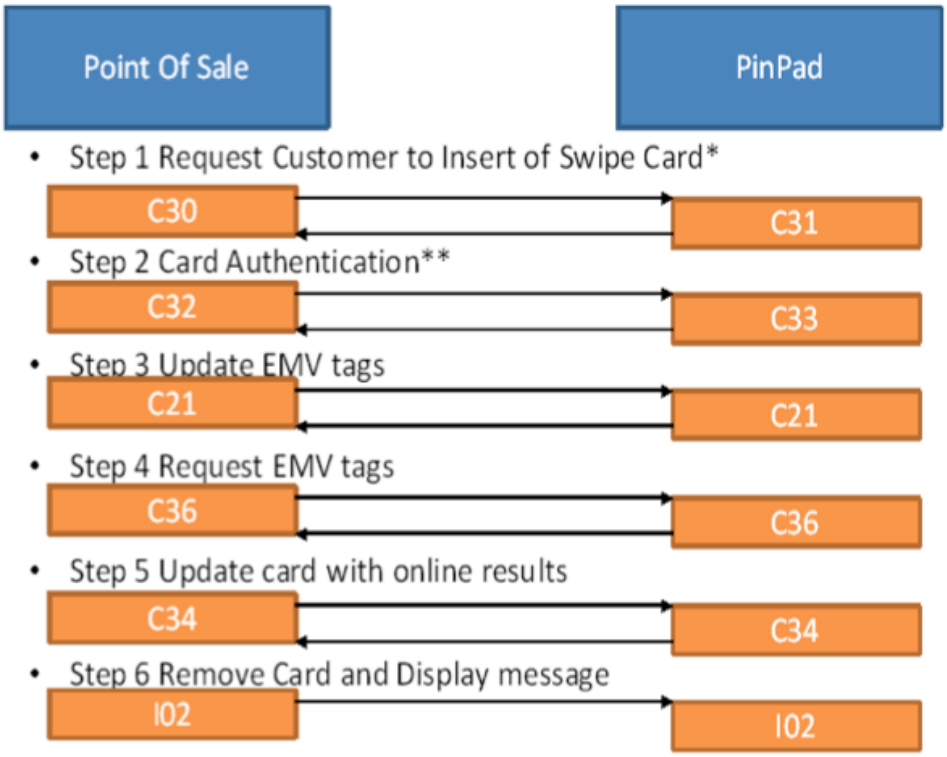
- Multiple D16, D15, D14, D17 and D18 commands will be sent for each card type supported
- 1) CTLSFloorLimit - Maximum amount for offline contactless transaction
- 2) CTLSCVMLimit - Amounts above this limit require signature
- 3) CTLSTRNLimit - Maximum amount allowed to be handled through contactless
- To support Visa opt out <modifyCandListFlag> needs to be set to one in D14 command.

5.9 Financial Transaction Commands

The following set of commands are the distinct set of individual methods that are available for use to directly interact with the PINPad for obtaining the card data and other necessary information from the card holder by which to send a financial transaction to the designated host processor for authorization. This section will be divided into two main areas. Those areas are the more granular command set for construction of a more customized transaction flow and a higher layer command set designed to simplify the development required to execute standard financial transaction flows.

EMV Transaction Flow

The prerequisite for performing EMV transactions is the initialization process defined in the Initialization Commands section of this document. Executing a complete transaction requires the use of multiple framework commands to interact with XPI to gather the appropriate customer data. The convenience functions (e.g. runPurchase) will execute the entire transaction flow utilizing pre-defined configuration and flow settings and handle these for all card types. The following flow (Figure 7) depicts the EMV (chip) transaction flow:



- Card request PDOL a U00 request will be sent from XPI. The PDOL is a list of tags and lengths of terminal-resident data elements needed by the ICC in processing the GET PROCESSING OPTIONS command. Only data elements having the terminal as the source of the data may be referenced in the PDOL.
- IF CVM results return online PIN required S67 command must be run to request PIN

EMV Tags

Numerous tags are returned from the chip card during an EMV transaction. Those tags are noted in the following tables segregated by the command which returns them.

Tag Identifier Returned from C30 Command	Description

4F	Application Identifier
9F12	Application preferred Name
50	Application Label
5F30	Service Code
5F20	Cardholder Name
57	Track 2 Equivalent Primary Account Number Field Separator (Hex 'D') Expiration Date (YYMM) Service Code Discretionary Data (defined by individual payment systems) Pad with one Hex 'F' if needed to ensure whole bytes
5A	Application Primary Account Number
5F24	Application Expiry Date
5F34	Application Primary account Number sequence number
5F25	Application Effective Date

Tag Identifier Returned from C32 Request	Description
C2	Signature Flag Authentication
95	Terminal Verification Results
9B	Transaction Status Information
9F27	Cryptogram Information Data
8A	Authorization response Code
9F26	Cryptogram returned by the ICC in response of the GENERATE AC command
9F36	Application Transaction Counter Counter maintained by the application in the ICC (incrementing the ATC is managed by the ICC)
9F10	Issuer Application Data

Tag Identifier C36 Additional Tags Requested	Description
9F41	Transaction Sequence Counter from terminal
9F02	Amount Authorized
9F03	Amount (other) (TIP)
5F36	Transaction currency component
9F1B	Terminal Floor Limit
9F1C	Terminal Identification
9F35	Terminal Type
9F1A	Terminal Country Code
5F2A	Transaction Currency Code
82	Application Interchange Profile Indicates the capabilities of the card to support specific functions in the application
9F37	Unpredictable Number Value to provide variability and uniqueness to the generation of a cryptogram
9C	Transaction Type
9F10	Issuer Application Data
84	Dedicated File Name

9F09	Application Version Number
9F34	CVM Cardholder verification Method
9F07	Application Usage Control Indicates issuer's specified restrictions on the geographic usage and services allowed for the application
9F0D	Issuer Action Code-Default
9F0E	Issuer Action Code - Denial
9F0F	Issuer Action Code - Online
9F33	Terminal Capabilities
5F34	Application PAN sequence Number
9F39	POS Entry Mode
8E	CVM List Identifies a method of verification of the cardholder supported by the application
5A	Application Primary Account Number
57	Track 2 equivalent
5F20 OR 9F0B	Cardholder name or Extended
9F1F	Track 2 discretionary data
9F7A	VLP process indicator
9F74	VLP authorization code

5.9.1 EMV Flow by Command

C30

- 1) ATR (Answer to Request)
- 2) Application Selection
- 3) Read Application Data

C32

- 1) Card Authentication
- 2) Processing Options
- 3) Terminal Risk Analysis
- 4) Cardholder verification
- 5) Terminal Action Analysis
- 6) Card Action Analysis
- 7) 1st Generate Application Cryptogram
- 8) Completion of CDA

C34

- 1) Script Processing Via C25 command
- 2) 2nd Generate Application Cryptogram
- 3) Completion Processing

Other XPI commands required depending on the transaction

- 1) S95 Request Serial Number
- 2) S21 Data Entry request

- 3) S67 Get encrypted PIN (online PIN request and PIN change)
- 4) C21 Update EMV Tags
- 5) C36 Get tag information
- 6) C25 Update Processor script (PIN change, PIN Unblock, Unblock Application, card)
  - Issuer either sends a 71 or 72 script, 71 is processed prior to the final Generate AC, and 72 is processed after the final generate AC. All scripts need to be initialized by sending a C34 to the PINPad.
- 7) I02 Remove Card and additional information about transaction status

5.10 Core Implementation using VMF

VeriFone Mobile Framework includes class libraries to interface with the VeriFone e210/e255/e315 Payment Device. This guide assume a fair understanding of Xcode 4.0+ and general Apple iOS programming knowledge.

5.10.1 Integrating with VMF

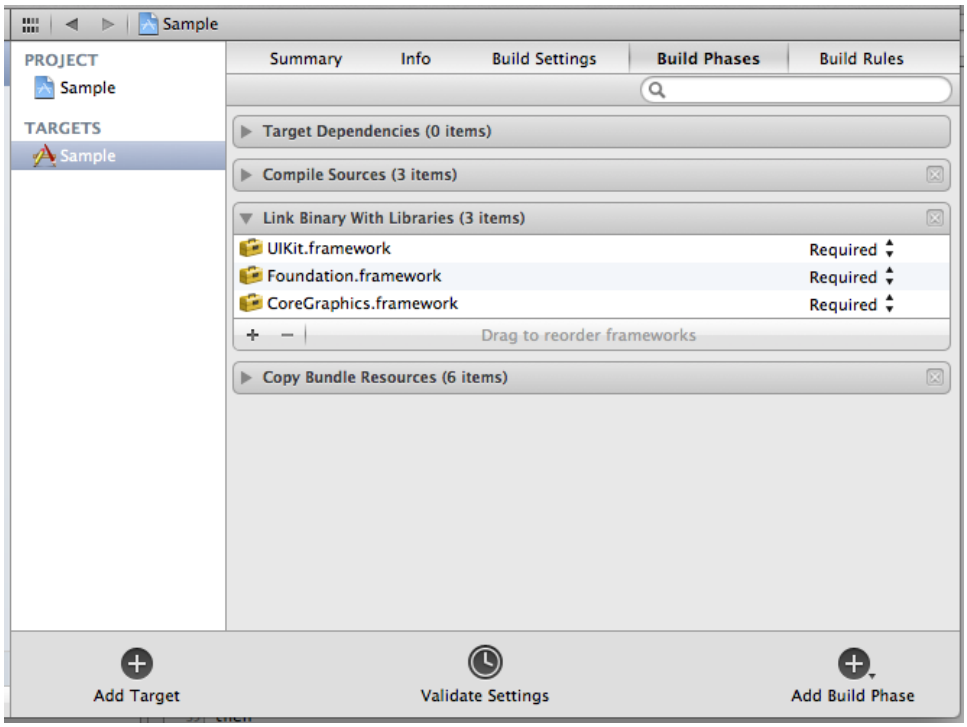
- Import the necessary framework/libraries
- Add Compiler Linker Flags
- Add Import statements to utilize frameworks
- Amend the view controller interface
- Implement optional delegate protocols
- Allocate/initialize e210/e255/e315 objects
- Add Protocol Strings to .pList
- Sample Project Tutorial

5.10.2 Import the necessary framework/libraries

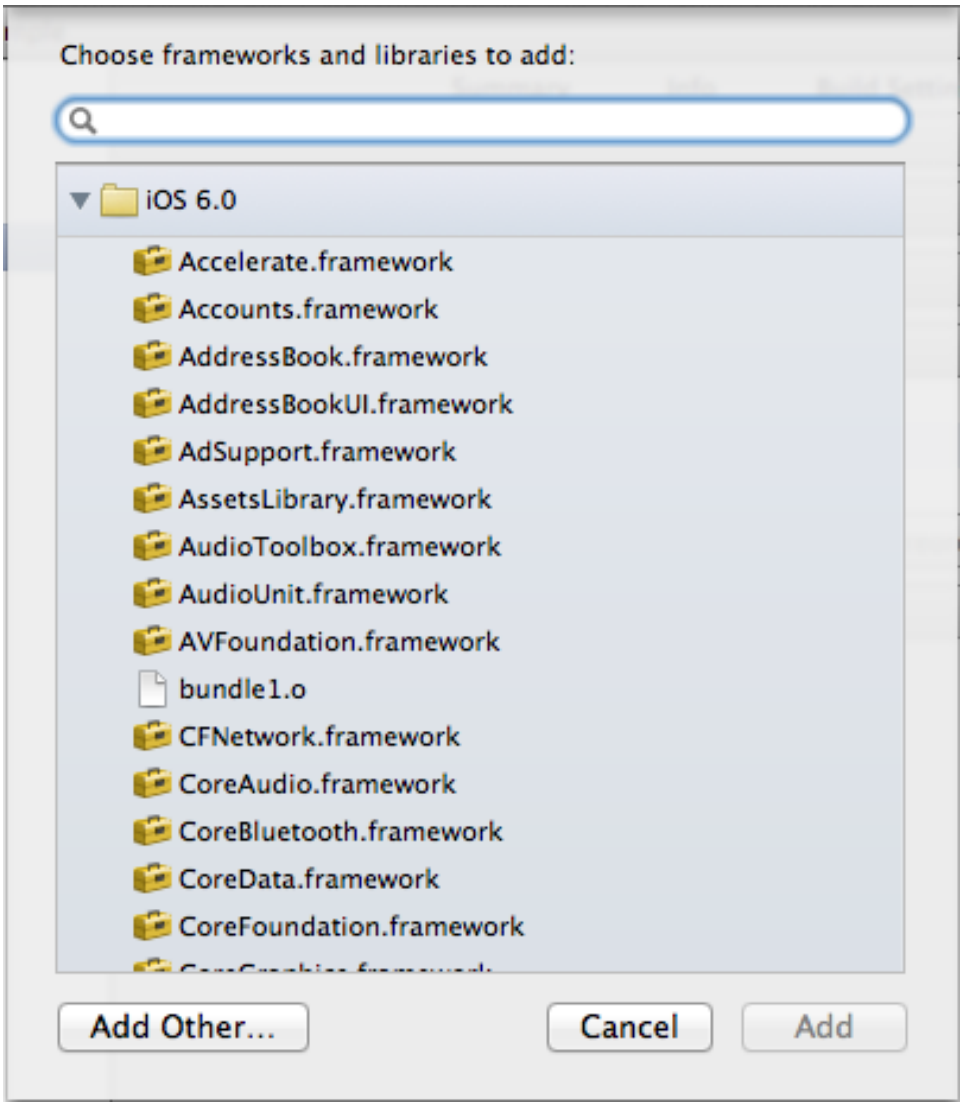
Communicating with the e210/e255/e315 requires the following framework/libraries to be imported into the project:

- VMF.framework
- libz.1.2.5.dylib
- ExternalAccessory.framework
- CFNetwork.framework

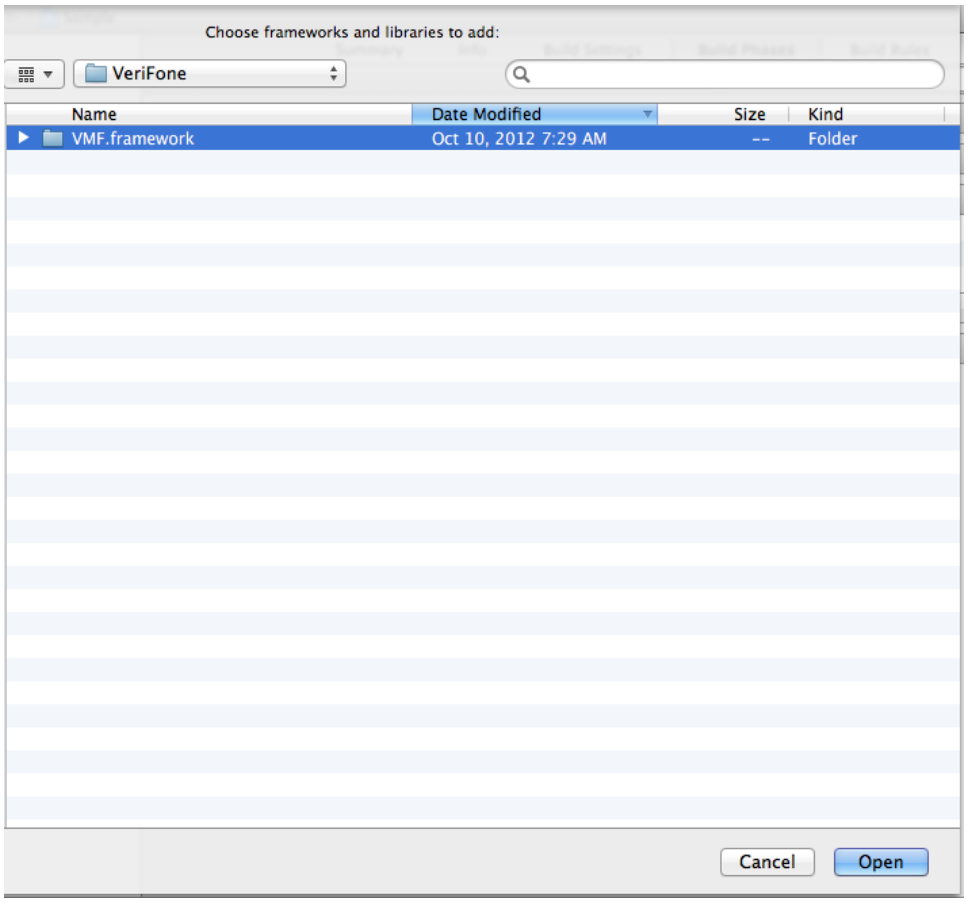
Under Build Phases, select Link Binary With Libraries and click the Add (+) button



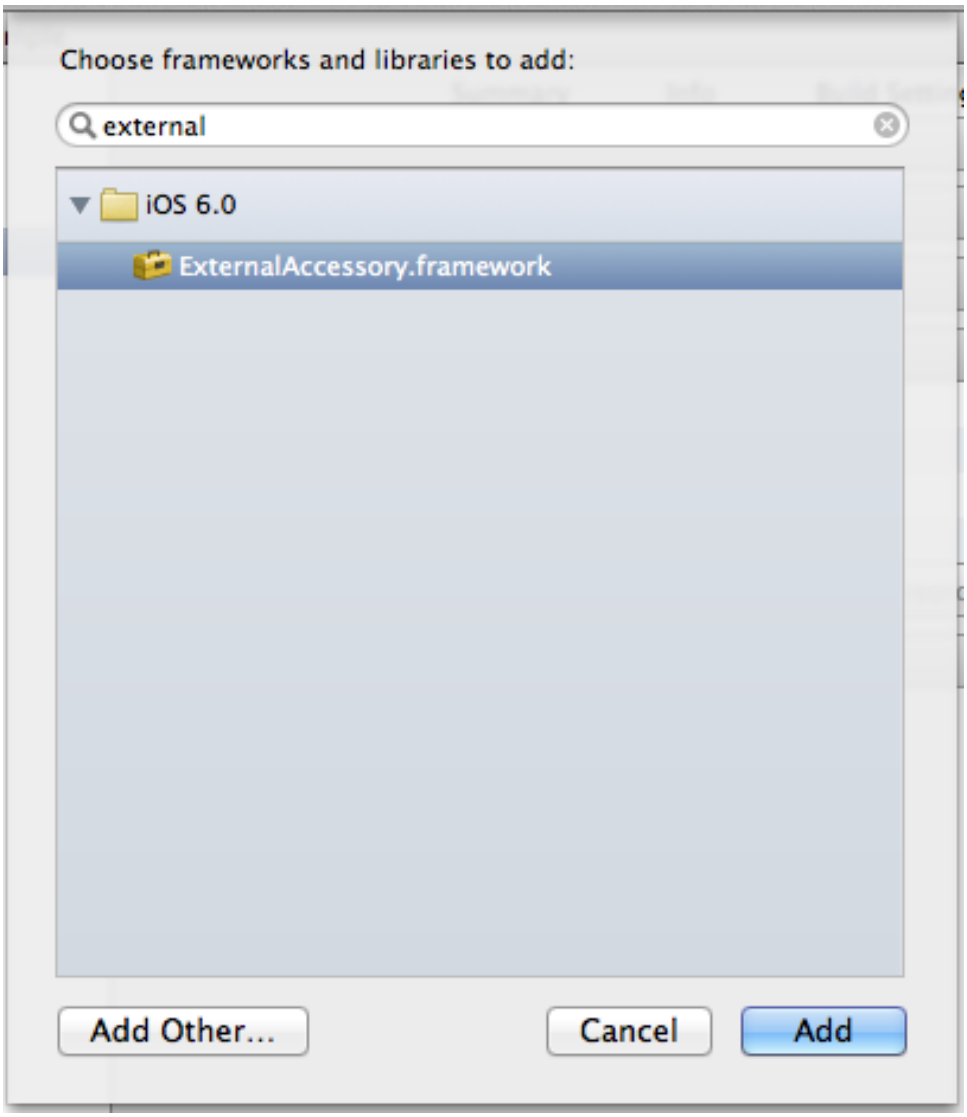
On the Choose Frameworks screen, click "Add Other" in the lower left



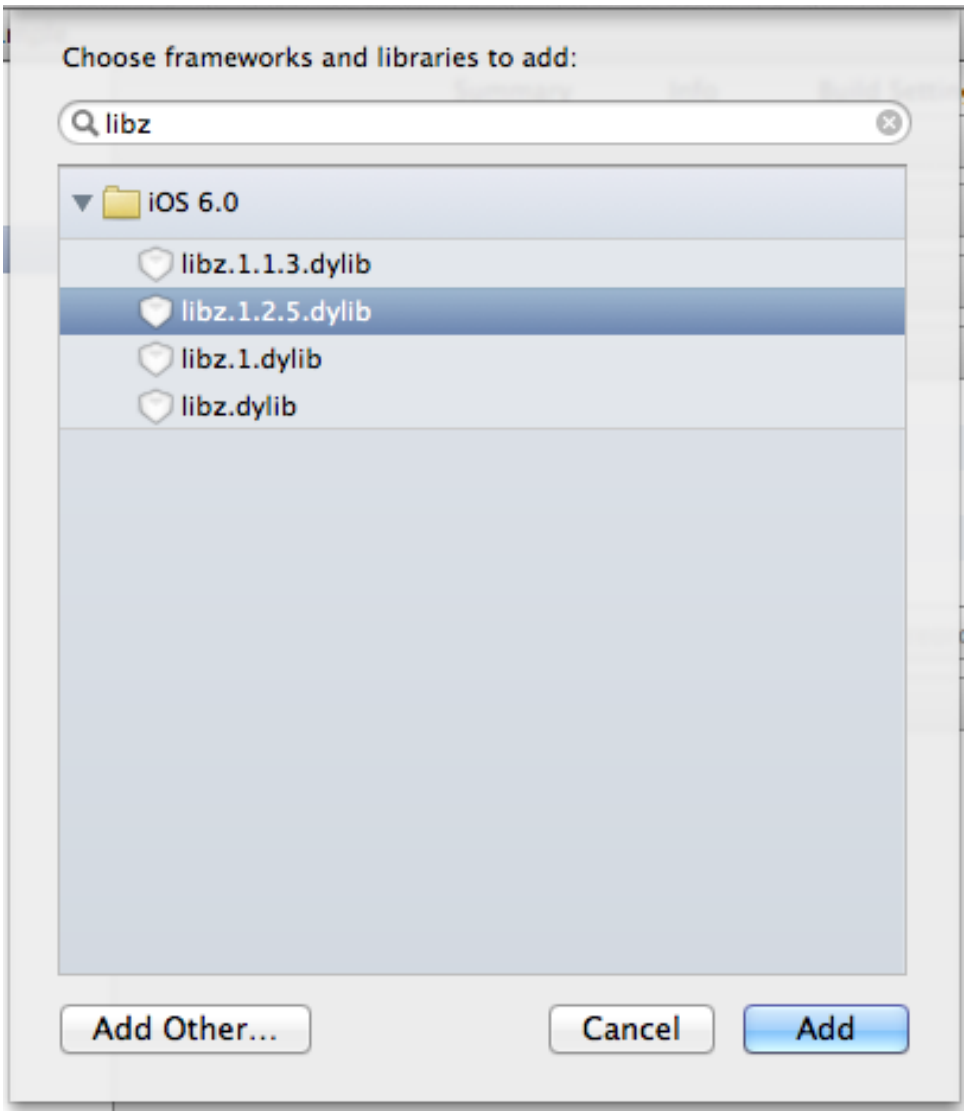
Navigate to the VMF.framework folder, and click "Open"



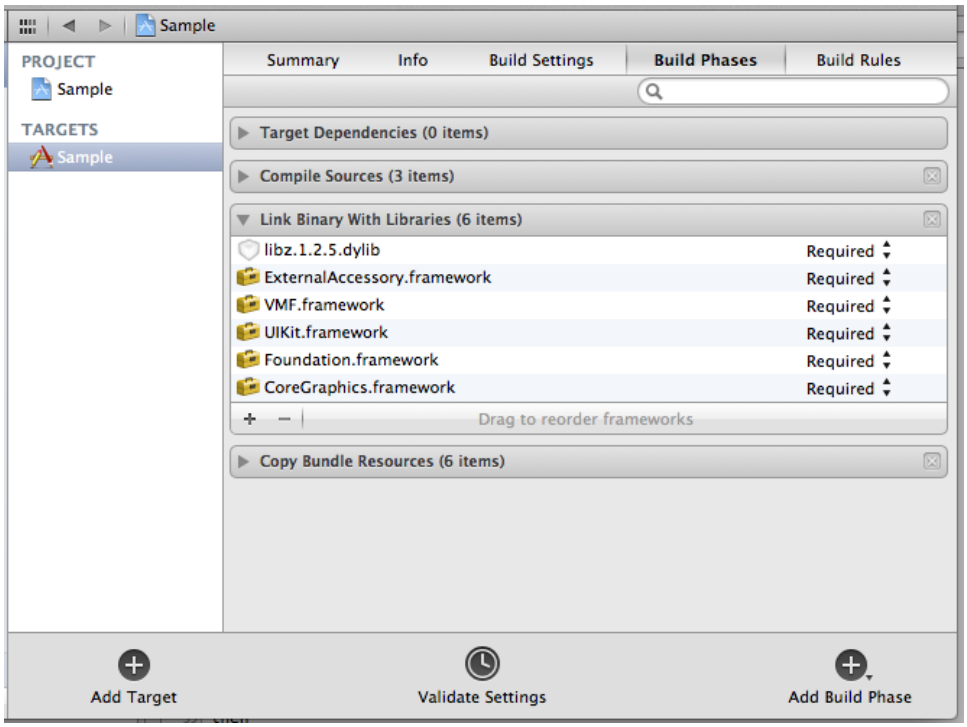
Link another library. On the Choose Frameworks screen, type "External" into the search bar, select ExternalAccesssory.-framework and click "Open"



Link another library. On the Choose Frameworks screen, type "libz" into the search bar, select libz.1.2.5.dylip and click "Open"



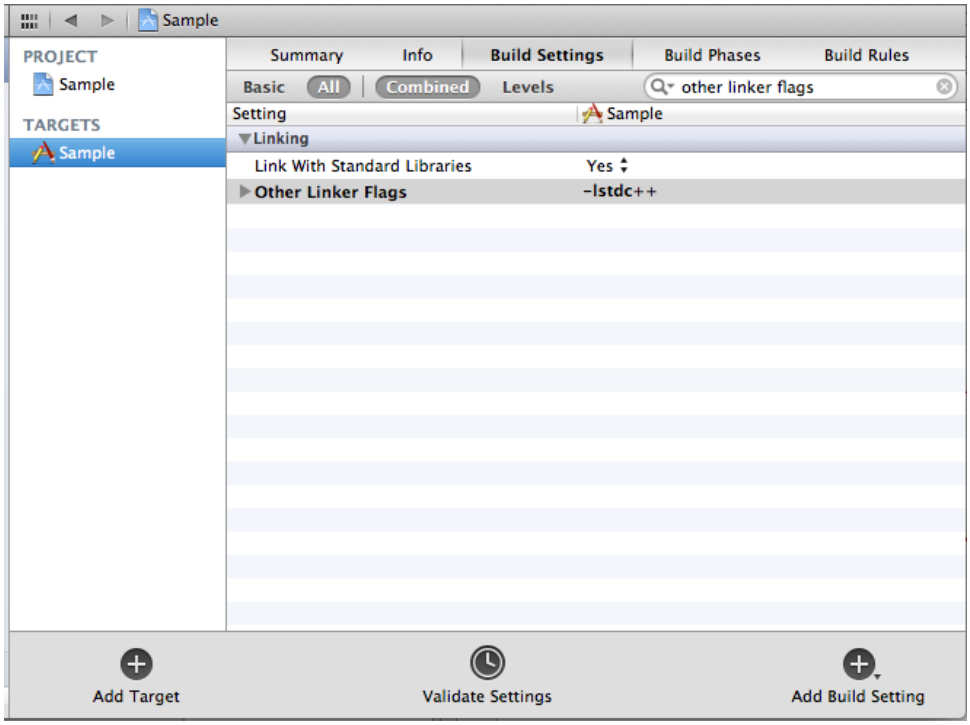
The Build Phases should now include the required frameworks/libraries for VMF



5.10.3 Add Compiler Linker Flags

In the XCode build settings, add the following flag under Other Linker Flags:

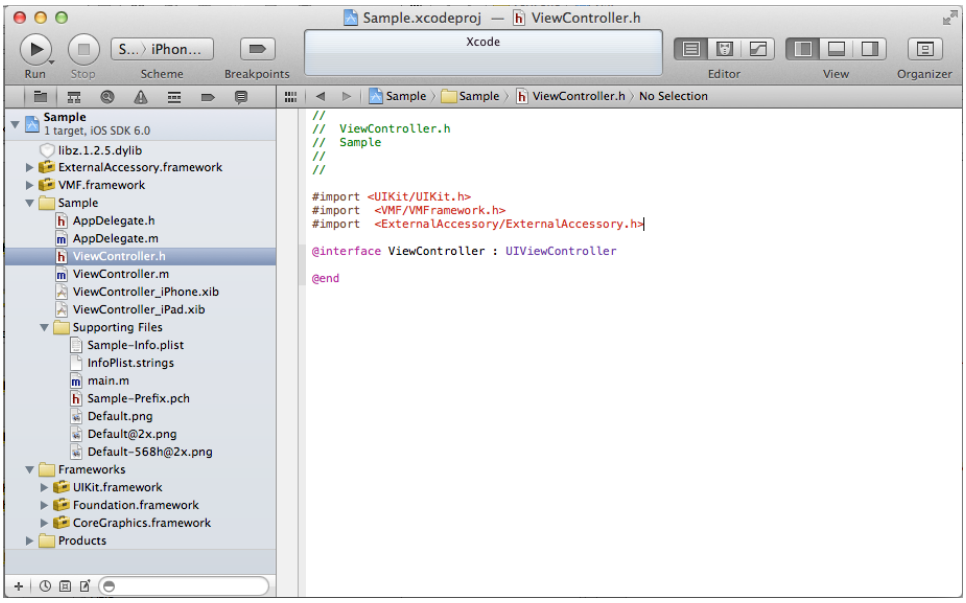
- `-lstdc++`



5.10.4 Add Import statements to utilize frameworks

In the header files of the classes that will access the e210/e255/e315, use import statement utilize the frameworks:

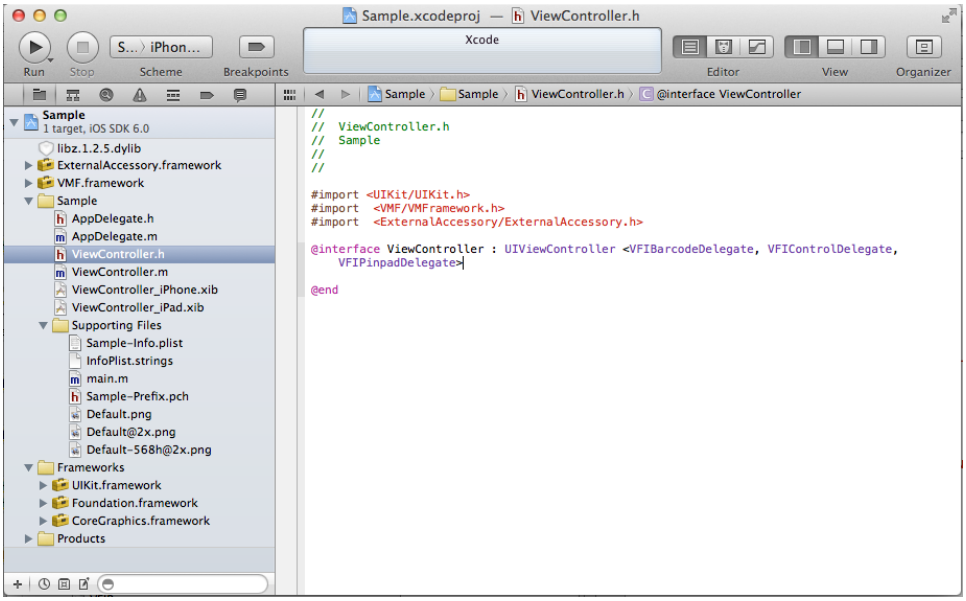
```
#import  <VMF/VMFramework.h>
#import  <ExternalAccessory/ExternalAccessory.h>
```



5.10.5 Amend the view controller interface to include the framework delegate classes:

In the header files of the classes that will be a delegate of VMF, include the reference to the framework delegate class name:

```
@interface Nav1ViewController: UIViewController <VFIBarcodeDelegate,
      VFIControlDelegate, VFIPinpadDelegate>
```



5.10.6 Implement any/all of the optional delegate protocols used to receive data from VMF:

VFIPinpadDelegate protocols

- pinpadResultCode:result: (VFIPinpadDelegate-p)
- pinpadLogEntry:withSeverity: (VFIPinpadDelegate-p)
- pinpadSerialData:incoming: (VFIPinpadDelegate-p)
- pinpadMSRData:expMonth:expYear:trackData: (VFIPinpadDelegate-p);
- pinpadMSRData:expMonth:expYear:track1Data:track2Data: (VFIPinpadDelegate-p);
- pinpadDownloadInfo: (VFIPinpadDelegate-p)log;
- pinpadDownloadBlocks:sent: (VFIPinpadDelegate-p);
- pinPadInitialized: (VFIPinpadDelegate-p)
- pinpadReconnectStarted (VFIPinpadDelegate-p)
- pinpadReconnectFinished (VFIPinpadDelegate-p)
- pinpadConnected: (VFIPinpadDelegate-p)
- pinpadDataReceived: (VFIPinpadDelegate-p)
- pinpadDataSent: (VFIPinpadDelegate-p)

```
- (void) pinpadResultCode:(NSString*)command result:(int)code{
}

- (void) pinpadLogEntry:(NSString*)logEntry withSeverity:(int)severity{
}

- (void) pinpadSerialData:(NSData*)data  incoming:(BOOL)isIncoming{
}

- (void) pinpadMSRData:(NSString*)pan  expMonth:(NSString*)month  expYear:(NSString*)year  trackData:(
    NSString*)track2{
}

- (void) pinpadMSRData:(NSString*)pan  expMonth:(NSString*)month  expYear:(NSString*)year  track1Data:(
    NSString*)track1  track2Data:(NSString*)track2{
}

- (void) pinpadDownloadInfo:(NSString*)log{
}

- (void) pinpadDownloadBlocks:(int)TotalBlocks sent:(int)BlocksSent{
}

- (void) pinPadInitialized:(BOOL)isInitialized{
}

- (void) lrcCalc:(int)reported calc:(int)calculated{
}

- (void) pinpadReconnectStarted{
}

- (void) pinpadReconnectFinished{
}

- (void) pinpadConnected:(BOOL)isConnected{
}

- (void) pinpadDataReceived:(NSData*)data{
}

- (void) pinpadDataSent:(NSData*)data{
}
```

VFIBarcodeDelegate protocols

- commandResult: (VFIBarcodeDelegate-p)
- barcodeTriggerEvent: (VFIBarcodeDelegate-p)
- barcodeLogEntry:withSeverity: (VFIBarcodeDelegate-p)
- barcodeSerialData:incoming: (VFIBarcodeDelegate-p)
- barcodeInitialized: (VFIBarcodeDelegate-p)
- barcodeReconnectStarted (VFIBarcodeDelegate-p)
- barcodeReconnectFinished (VFIBarcodeDelegate-p)
- barcodeConnected: (VFIBarcodeDelegate-p)
- barcodeDataReceived: (VFIBarcodeDelegate-p)
- barcodeDataSent: (VFIBarcodeDelegate-p)
- barcodeScanData:barcodeType: (VFIBarcodeDelegate-p)

```
- (void) commandResult:(int)result{
}

- (void) barcodeTriggerEvent:(int)BCS_TRIGGER{
}

- (void) barcodeLogEntry:(NSString*)logEntry withSeverity:(int)severity{
}

- (void) barcodeSerialData:(NSData*)data  incoming:(BOOL)isIncoming{
}

- (void) barcodeInitialized:(BOOL)isInitialized{
}

- (void) barcodeReconnectStarted{
}

- (void) barcodeReconnectFinished{
}

- (void) barcodeConnected:(BOOL)isConnected{
}

- (void) barcodeDataReceived:(NSData*)data{
}

- (void) barcodeDataSent:(NSData*)data{
}

- (void) barcodeScanData:(NSData*)data barcodeType:(int)thetype{
}
```

VFIControlDelegate protocols

- controlLogEntry:withSeverity: (VFIControlDelegate-p)
- controlInitialized: (VFIControlDelegate-p)
- controlReconnectStarted (VFIControlDelegate-p)
- controlReconnectFinished (VFIControlDelegate-p)
- controlConnected: (VFIControlDelegate-p)

- controlDataReceived: (VFIControlDelegate-p)
- controlDataSent: (VFIControlDelegate-p)

```
- (void) controlLogEntry:(NSString*)logEntry withSeverity:(int)severity{
}

- (void) controlInitialized:(BOOL)isInitialized{
}

- (void) controlReconnectStarted{
}

- (void) controlReconnectFinished{
}

- (void) controlConnected:(BOOL)isConnected{
}

- (void) controlDataReceived:(NSData*)data{
}

- (void) controlDataSent:(NSData*)data{
}
```

5.10.7 Allocate and initialize the e210/e255/e315 framework objects:

To use the e210/e255/e315, you must first initialize the framework, then set your class as a delegate to receive messages through any desired optional protocols, then you must initialize the application(s) on the e210/e255/e315.

Direct initialization of the framework by a class in your project is as follows:

```
VFIPinpad *pinPad = [[VFIPinpad alloc] init];
[pinPad setDelegate:self];
[pinPad initDevice];

VFIBarcode *barcode:[[VFIBarcode alloc] init];
[barcode setDelegate:self];
[barcode initDevice];

VFIControl *control:[[VFIControl alloc] init];
[control setDelegate:self];
[control initDevice];
```

It is Best Practice to establish a Singleton Class to initialize the framework. Please see Establish a Singleton Class to access e210/e255/e315. Establishing a Singleton Class called VFDevice (in our example) would then require the following implementation to utilize the e210/e255/e315:

```
[[VFDevice pinPad] setDelegate:self];
[[VFDevice pinPad] initDevice];

[[VFDevice barcode] setDelegate:self];
[[VFDevice barcode] initDevice];

[[VFDevice control] setDelegate:self];
[[VFDevice control] initDevice];
```

5.10.8 Add the e210/e255/e315 device protocol strings to your projects .pList:

Add all off the protocol strings to your application .pList under "Supported external accessory protocols" array:

e210 Dock Connector

- com.verifone.pmr2.control
- com.verifone.pmr2.zontalk

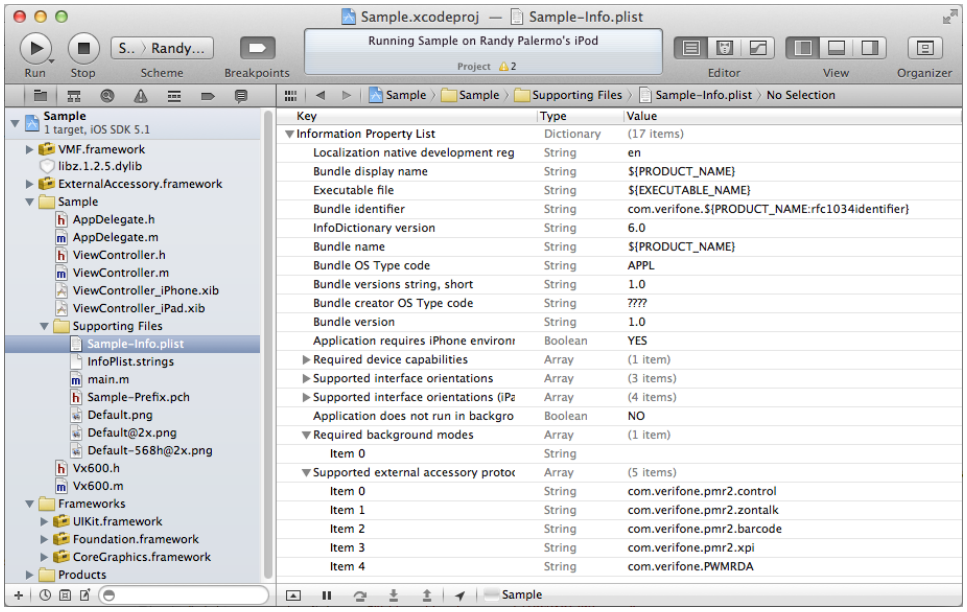
- com.verifone.pmr2.barcode
- com.verifone.pmr2.xpi

e255 Bluetooth

- com.verifone.PWMRDA

e315 Lightning Connector

- com.verifone.pmr.control
- com.verifone.pmr.zontalk
- com.verifone.pmr.barcode
- com.verifone.pmr.xpi



5.10.9 Sample Project Tutorial

Using Xcode 5.0+, we will create a sample project that will interface with the e210/e255/e315/e335 and will perform the following activities:

Control Functions:

- Turn On/Off Keypad Power
- Turn On/Off Keypad Beep

Barcode Functions:

- Turn On/Off Scanner
- Set 1D/2D Scanning Mode
- Set Scanner Control Mode (Edge, Level, Soft, Passive)

- Send Trigger Event (for Soft and Passive modes)

Pinpad Functions:

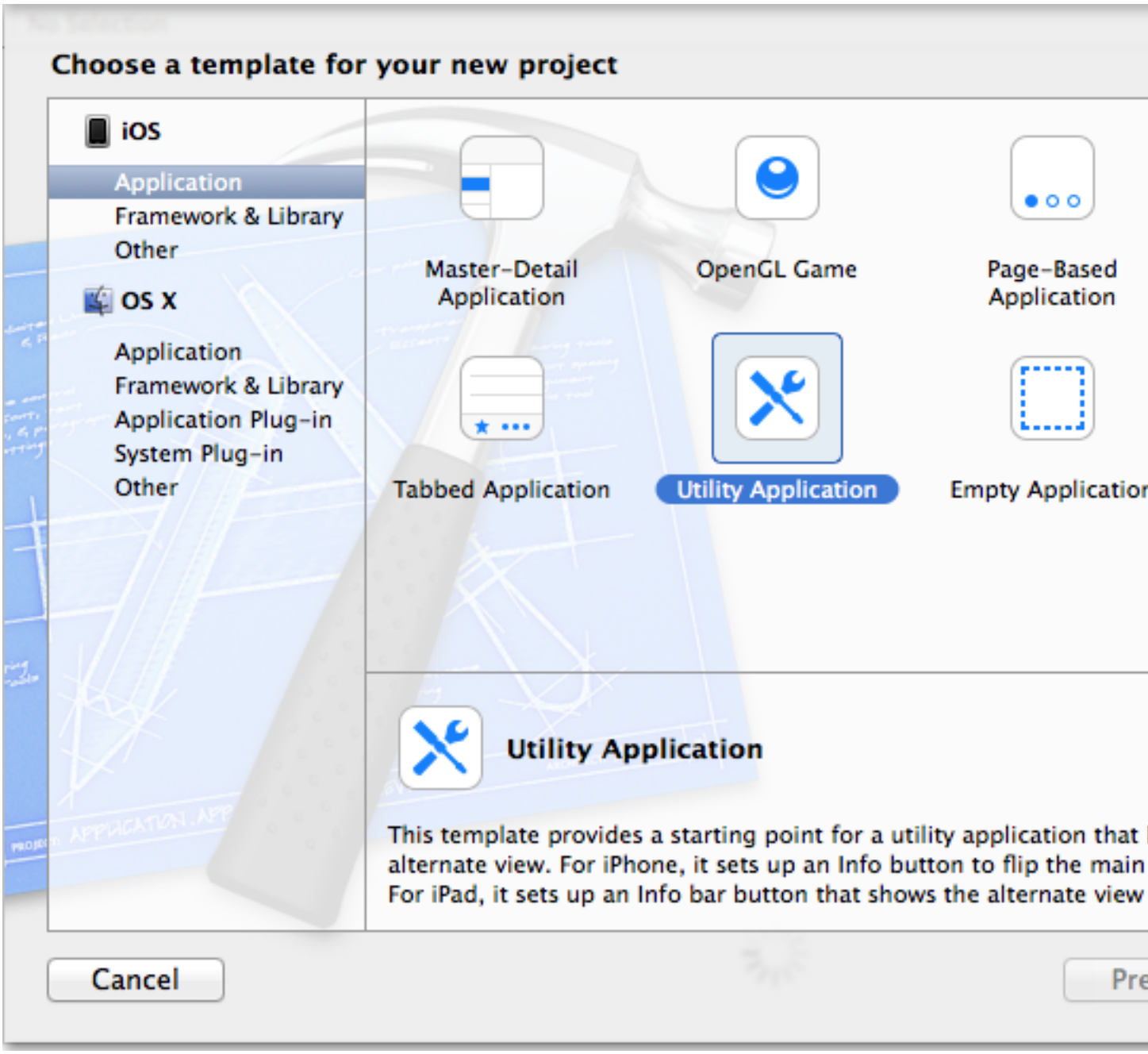
- Turn On/Off MSR for unsolicited swipes
- Perform a full sample Credit/PIN-based Debit transaction complete with PIN Entry
- Perform Sample PIN Entry
- Perform Sample Request swipe entry
- Prompt User to confirm various messages

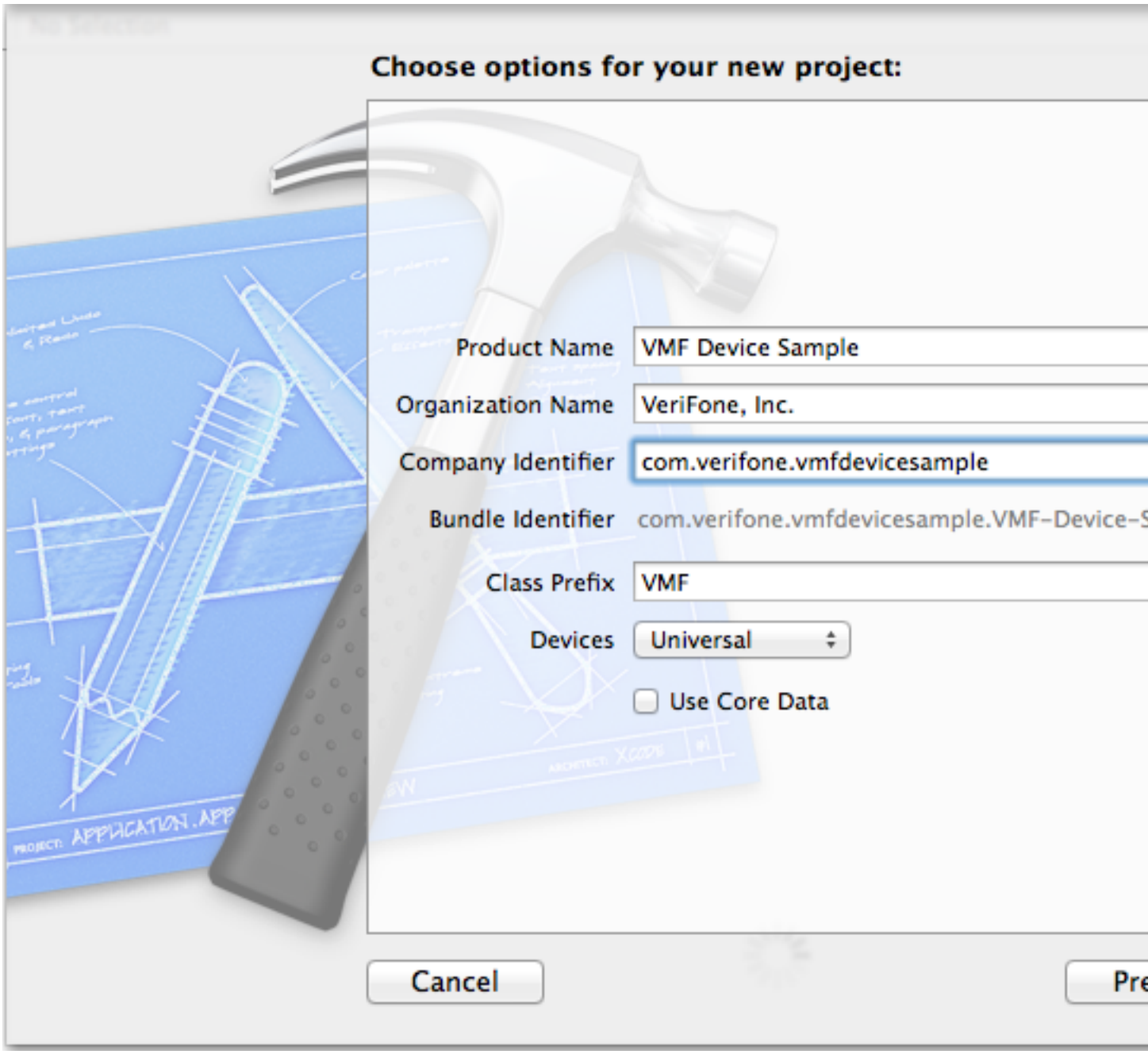
Protocol Delegates:

- Barcode Protocol to report barcode scan data
- Barcode Protocol to report connect/disconnect events
- Pinpad Protocol to report unsolicited card swipes
- Pinpad Protocol to report connect/disconnect events
- Control Protocol to report connect/disconnect events

5.10.9.1 Step 1: Create New Project

Create a new Utility Application in Xcode


Choose a template for your new project
iOS
Application
Framework & Library
Other
OS X
Application
Framework & Library
Application Plug-in
System Plug-in
Other
Master-Detail Application
OpenGL Game
Page-Based Application
Tabbed Application
Utility Application
Empty Application
Utility Application
This template provides a starting point for a utility application that
alternate view. For iPhone, it sets up an Info button to flip the main
For iPad, it sets up an Info bar button that shows the alternate view
Cancel



5.10.9.2 Step 2: Import Frameworks

Import the necessary framework/libraries

5.10.9.3 Step 3: Add Compiler Linker Flags

Add Compiler Linker Flags

5.10.9.4 Step 4: Add Protocol Strings to .pList

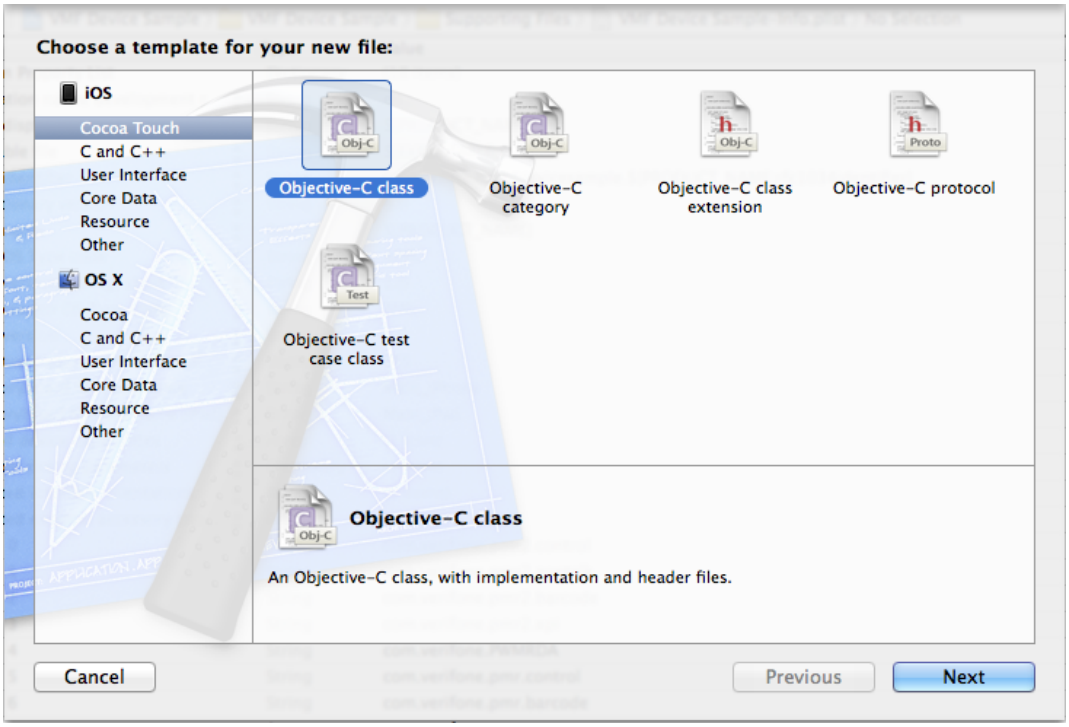
We would like the application to work with Bluetooth and non-Bluetooth versions of e210, e255,e315,e335, so we will add all 9 protocol strings to the .pList:

Add Protocol Strings to .pList

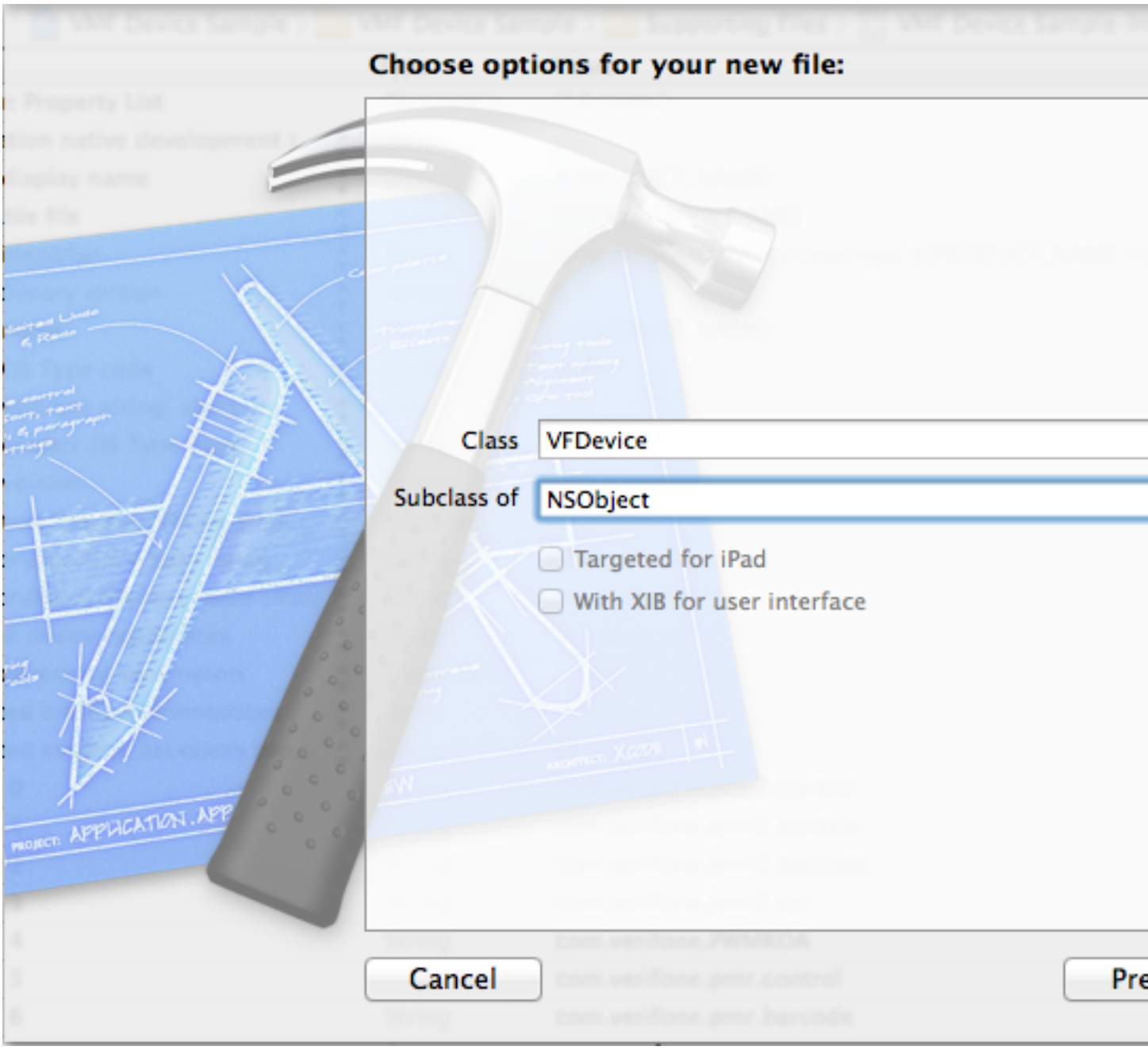
5.10.9.5 Step 5: Create e210/e255/e315/e335 Singleton Class

This step adds a Singleton class object to the project. You may call this anything you like.

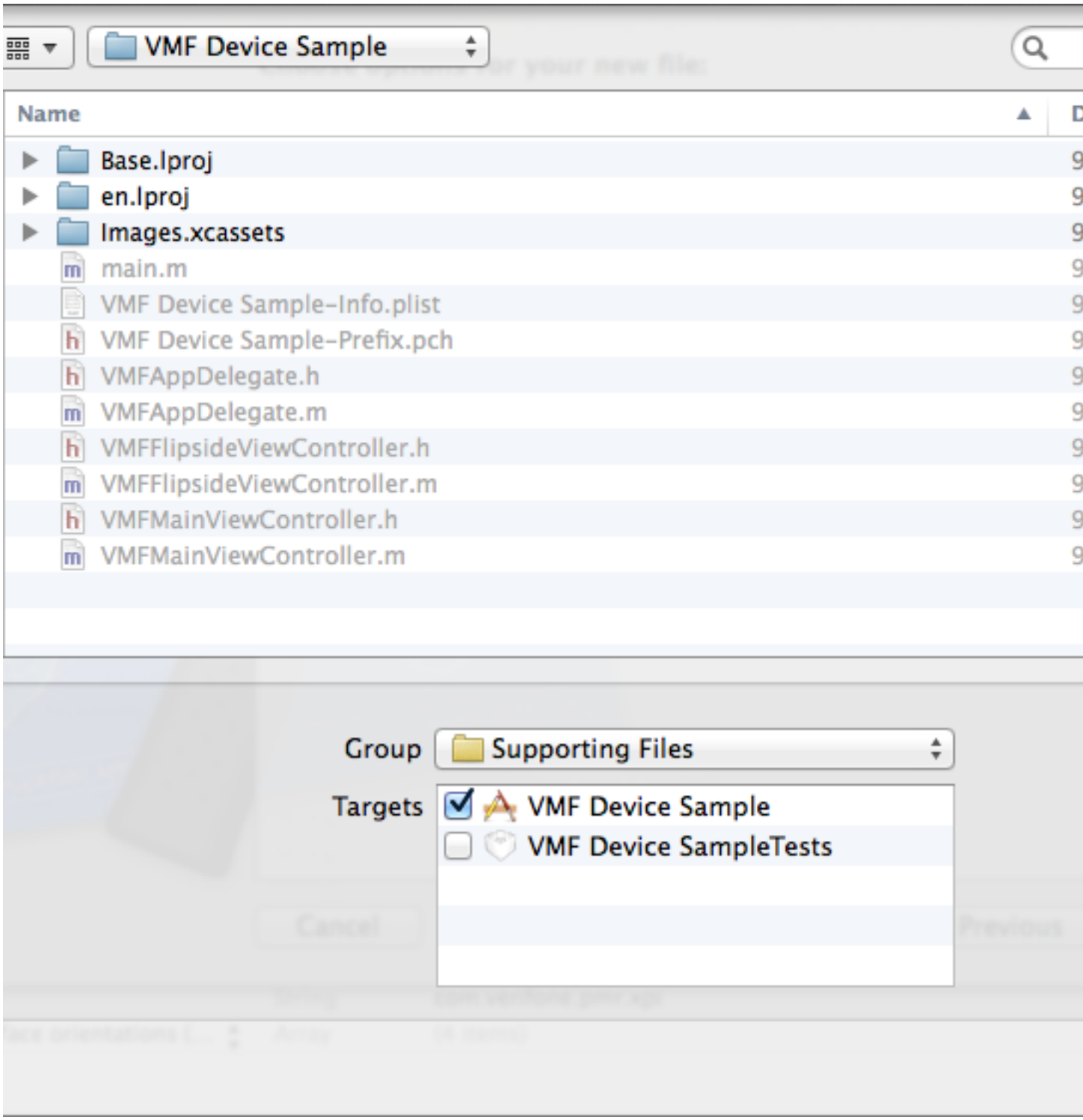
Add a new file to your project. Select “Objective-C” class under “iOS Cocoa Touch” and click “Next”.



Name the class “VFDevice” and set the Subclass to “NSObject” and click “Next”.

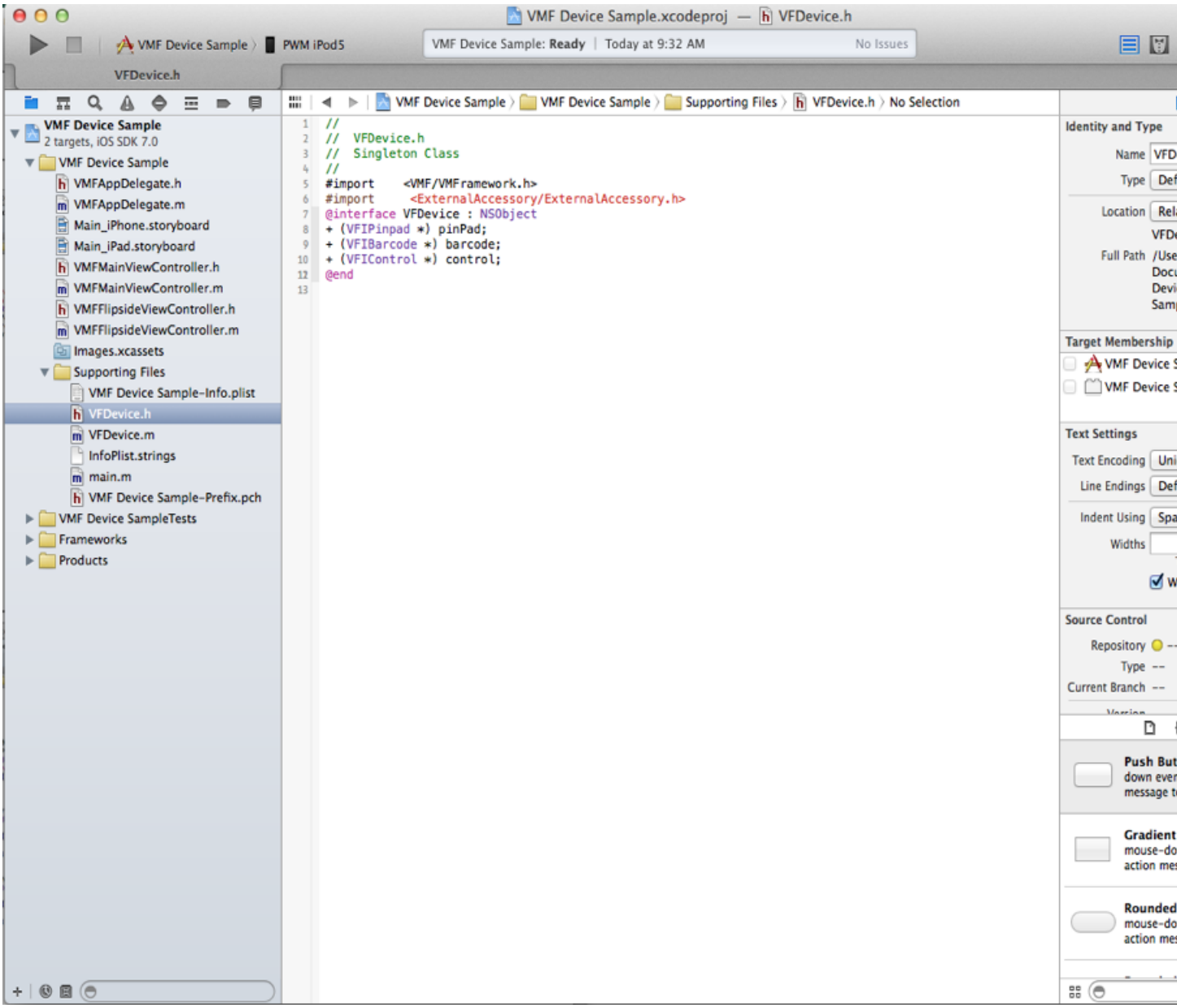


Make sure your project target is selected in the bottom of the save dialog box. Save the class file in your project folder by pressing "Create"



This will give you a blank class file named VFDevice.

Please reference Establish a Singleton Class to access e210/e255/e315 and copy that code into your blank VFDevice class.


VMF Device Sample.xcodeproj — VFDevice.h
VMF Device Sample
PWM iPod5
VMF Device Sample: Ready
Today at 9:32 AM
No Issues
VFDevice.h
VMF Device Sample
2 targets, iOS SDK 7.0
VMF Device Sample
VMFAppDelegate.h
VMFAppDelegate.m
Main_iPhone.storyboard
Main_iPad.storyboard
VMFMainViewController.h
VMFMainViewController.m
VMFFlipsideViewController.h
VMFFlipsideViewController.m
Images.xcassets
Supporting Files
VMF Device Sample-Info.plist
VFDevice.h
VFDevice.m
InfoPlist.strings
main.m
VMF Device Sample-Prefix.pch
VMF Device SampleTests
Frameworks
Products
VMF Device Sample
VMF Device Sample
Supporting Files
VFDevice.h
No Selection
//
// VFDevice.h
// Singleton Class
//
#import <VMF/VMFramework.h>
#import <ExternalAccessory/ExternalAccessory.h>
@interface VFDevice : NSObject
+ (VFIPinpad *) pinPad;
+ (VFIBarcode *) barcode;
+ (VFIControl *) control;
@end
Identity and Type
Name
Type
Location
Full Path
Target Membership
Text Settings
Text Encoding
Line Endings
Indent Using
Widths
Source Control
Repository
Type
Current Branch

```
//
//  VFDevice.m
//  VMF Device Sample
//
//  Created by James Gnann on 1/10/14.
//  Copyright (c) 2014 VeriFone, Inc. All rights reserved.
//

#import "VFDevice.h"

@implementation VFDevice
+ (VFIPinpad *) pinPad{
    static VFIPinpad *_pinPad = nil;
    if (_pinPad == nil) {
        _pinPad = [[VFIPinpad alloc] init];
    }
    return _pinPad;
}
+ (VFIBarcode *) barcode{
    static VFIBarcode *_barcode = nil;
    if (_barcode == nil) {
        _barcode = [[VFIBarcode alloc] init];
    }
    return _barcode;
}
+ (VFIControl *) control{
    static VFIControl *_control = nil;
    if (_control == nil) {
        _control = [[VFIControl alloc] init];
    }
    return _control;
}
@end

```

5.10.9.6 Step 6: Design Interface

Design the User Interface by editing the Main_iPhone/iPad .storyboard file.

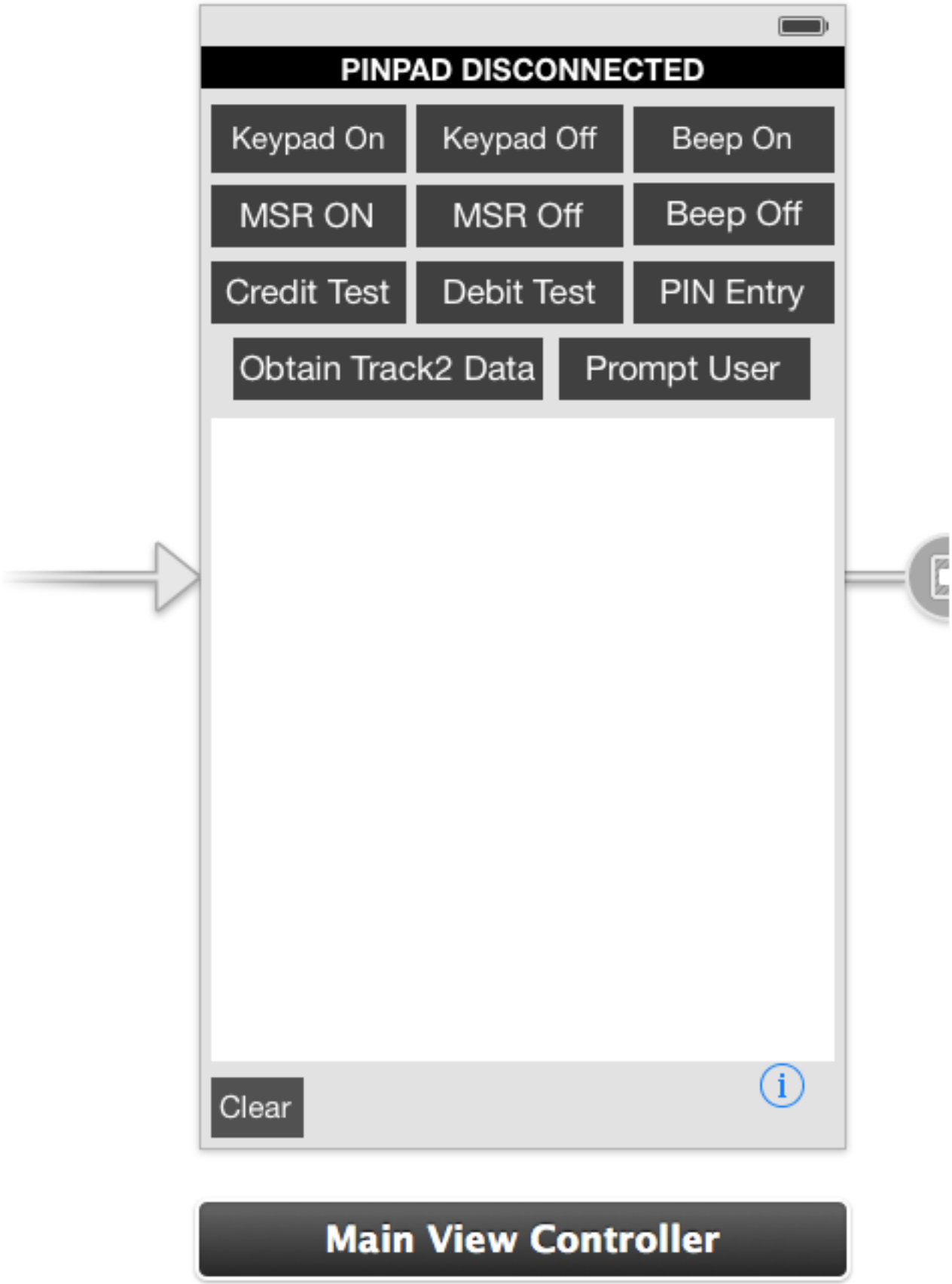
Make sure that your storyboard contains the following buttons and fields on the Main View Controller:

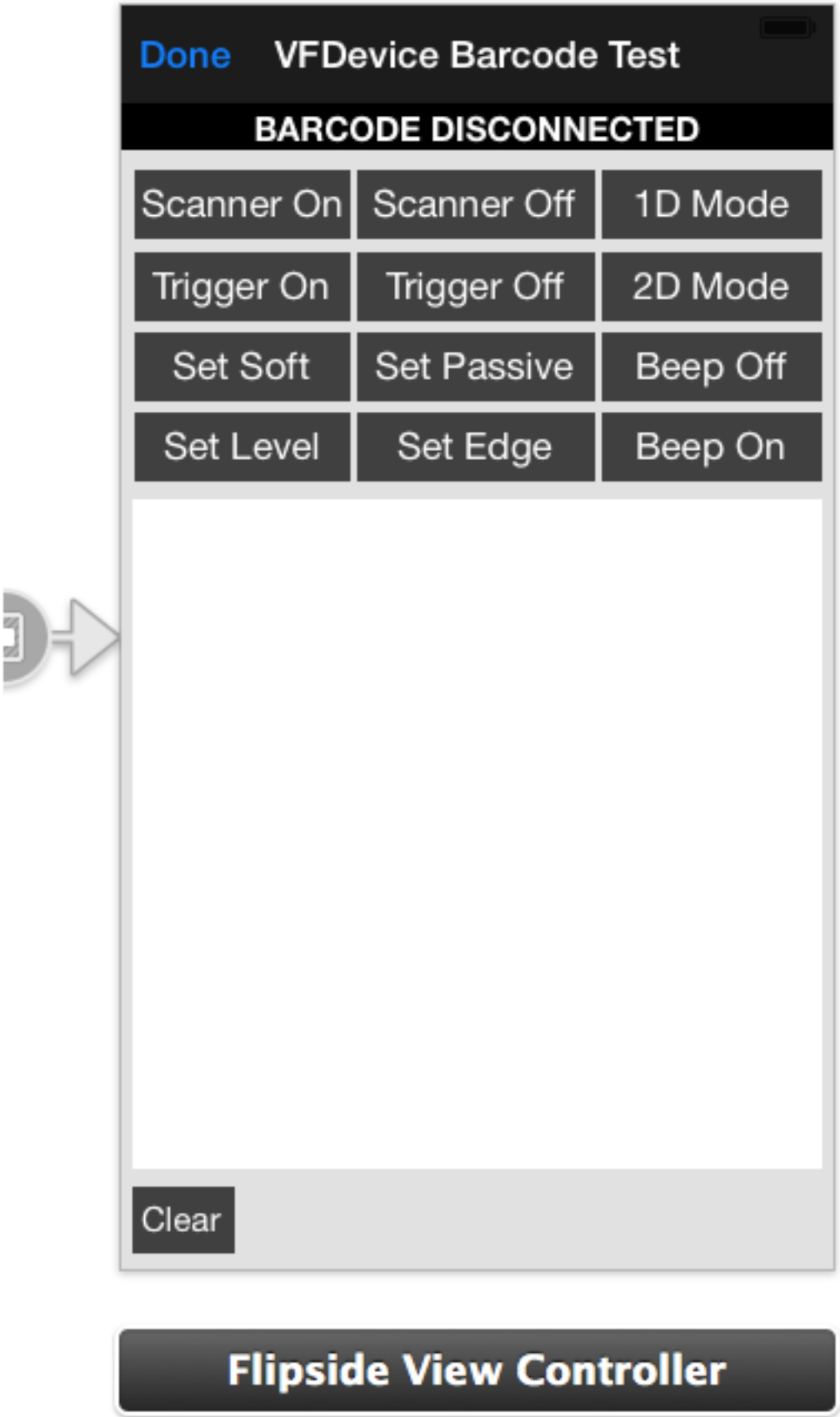
- Add a label to the top that will signify connection/disconnection status of the various parts of the device. In the provided example, it is 15pt Bold, centered white text on a black background.
- Add a blank text view to communicate data from the e210/e255/e315/e335. Remove the Editable behavior if you don't want the keyboard to pop up if you select it.
- Add buttons to execute the following commands. In the example, they are Default buttons with white text on Dark Gray background with an inset of 3.

  - Keypad Power ON
  - Keypad Power OFF
  - Keypad Beep ON
  - Keypad Beep OFF
  - MSR ON
  - MSR OFF
  - Credit Transaction
  - Debit Transaction
  - Prompt User
  - PIN Entry
  - Obtain Track2 Data
  - Clear Text View

Make sure that your storyboard contains the following buttons and fields on the Flipside View Controller:

- Add a label to the top that will signify connection/disconnection status of the various parts of the device. In the provided example, it is 15pt Bold, centered white text on a black background.
- Add a blank text view to communicate data from the e210/e255/e315/e335. Remove the Editable behavior if you don't want the keyboard to pop up if you select it.
- Add buttons to execute the following commands. In the example, they are Default buttons with white text on Dark Gray background with an inset of 3.
  - Barcode Scanner ON
  - Barcode Scanner OFF
  - Barcode Scanner 1D
  - Barcode Scanner 2D
  - Barcode Scanner Trigger ON
  - Barcode Scanner Trigger OFF
  - Barcode Scanner Level Mode
  - Barcode Scanner Edge Mode
  - Barcode Scanner Soft Mode
  - Barcode Scanner Passive Mode
  - Barcode Scanner Beep After Scan ON
  - Barcode Scanner Beep After Scan OFF
  - Clear Text View

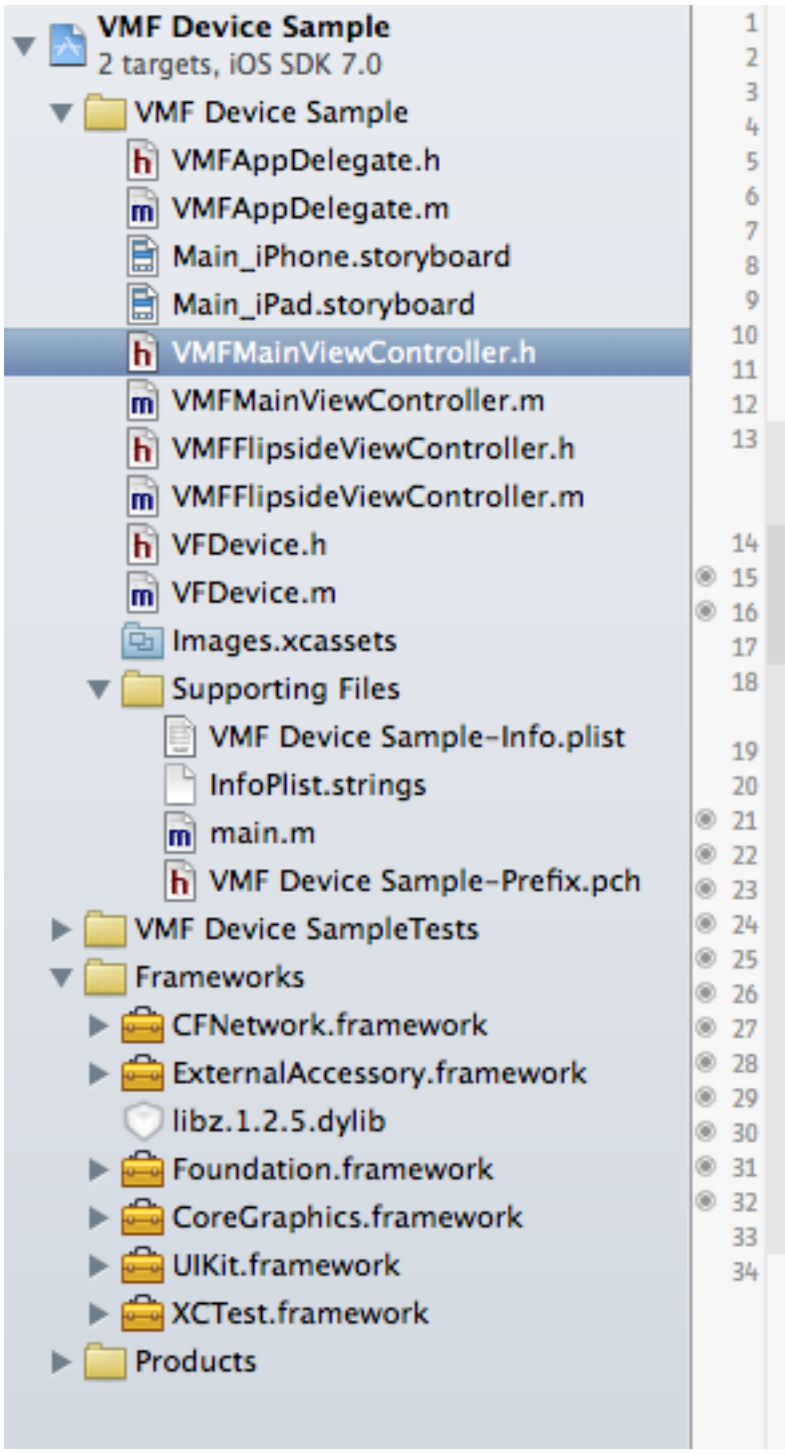

PINPAD DISCONNECTED
Keypad On
Keypad Off
Beep On
MSR ON
MSR Off
Beep Off
Credit Test
Debit Test
PIN Entry
Obtain Track2 Data
Prompt User
Clear
Main View Controller

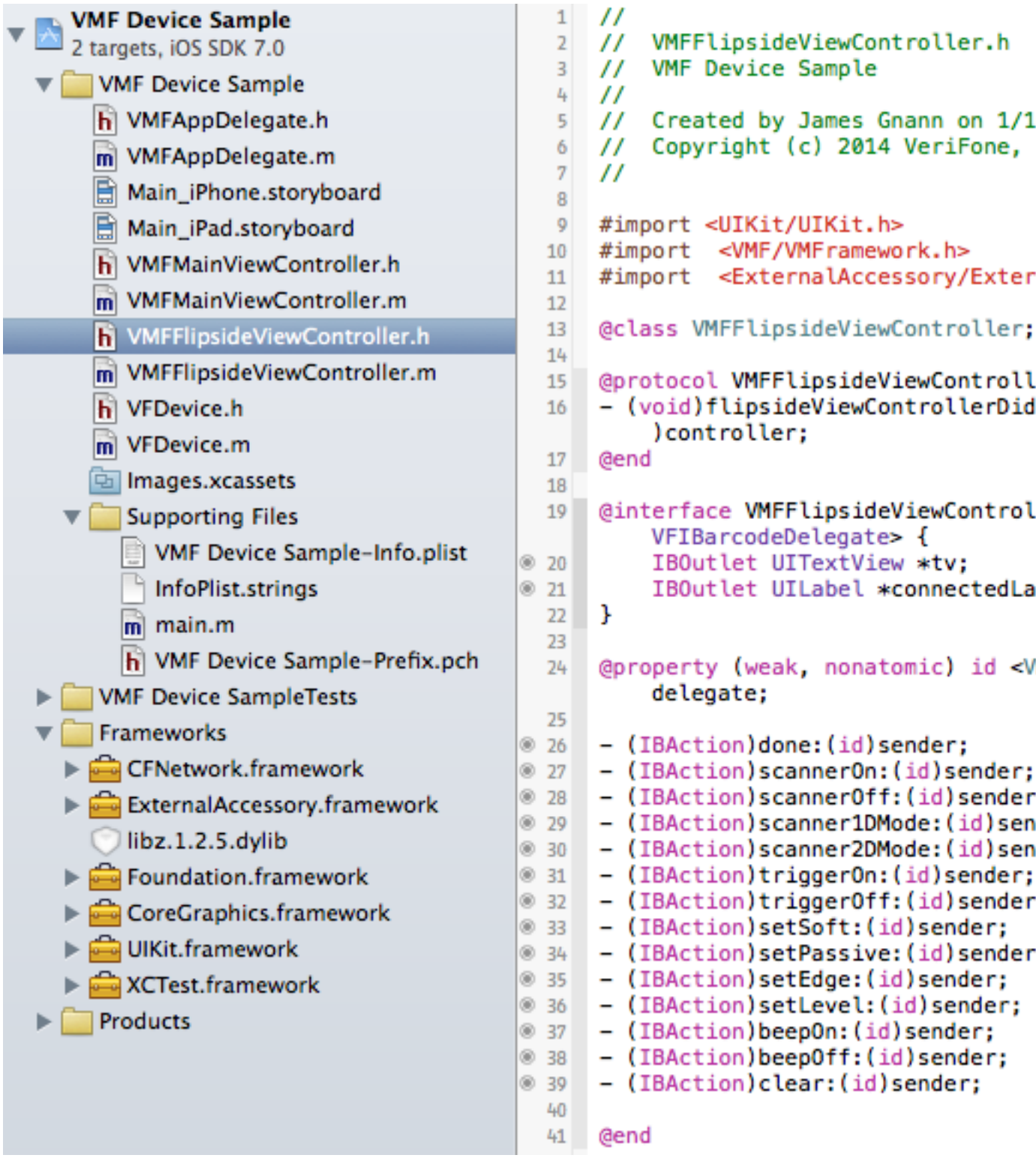

Done
VFDevice Barcode Test
BARCODE DISCONNECTED
Scanner On
Scanner Off
1D Mode
Trigger On
Trigger Off
2D Mode
Set Soft
Set Passive
Beep Off
Set Level
Set Edge
Beep On
Clear
Flipside View Controller

5.10.9.7 Step 7: Configure Header File

In the header file, perform the following:

- Add Import statements to utilize frameworks
- Amend the view controller interface
- Create an IBOutlet for the UITextView and link it as a Referencing Outlet to the UITextView on the storyboard
- Create an IBOutlet for the UILabel and link it as a Referencing Outlet to the UILabel on the storyboard
- Create the 25 IBAction for the buttons and link them to the “Touch Up Inside” event on the storyboard buttons.


VMF Device Sample
2 targets, iOS SDK 7.0
VMF Device Sample
VMFAppDelegate.h
VMFAppDelegate.m
Main_iPhone.storyboard
Main_iPad.storyboard
VMFMainViewController.h
VMFMainViewController.m
VMFFlipsideViewController.h
VMFFlipsideViewController.m
VFDevice.h
VFDevice.m
Images.xcassets
Supporting Files
VMF Device Sample-Info.plist
InfoPlist.strings
main.m
VMF Device Sample-Prefix.pch
VMF Device SampleTests
Frameworks
CFNetwork.framework
ExternalAccessory.framework
libz.1.2.5.dylib
Foundation.framework
CoreGraphics.framework
UIKit.framework
XCTest.framework
Products
//
// VMFMainViewController.h
// VMF Device Sample
//
// Created by James Gnann on 1/1
// Copyright (c) 2014 VeriFone,
//
#import "VMFFlipsideViewControlle
#import <VMF/VMFramework.h>
#import <ExternalAccessory/Exter
@interface VMFMainViewController
VMFFlipsideViewControllerDele
VFIControlDelegate, VFIPinpad
{
IBOutlet UITextView *tv;
IBOutlet UILabel *connectedLa
}
@property (strong, nonatomic) UIP
flipsidePopoverController;
-(IBAction) keypadOn:(id)sender;
-(IBAction) keypadOff:(id)sender;
-(IBAction) beepOn:(id)sender;
-(IBAction) beepOff:(id)sender;
-(IBAction) MSROn:(id)sender;
-(IBAction) MSROff:(id)sender;
-(IBAction) creditTest:(id)sender
-(IBAction) debitTest:(id)sender;
-(IBAction) PINEntryTest:(id)send
-(IBAction) obtainTrackData:(id)s
-(IBAction) promptUser:(id)sender
-(IBAction) clear:(id)sender;
@end



VMFMainViewController.h:

```
//
//  VMFMainViewController.h
//  VMF Device Sample
```

```
#import "VMFFlipsideViewController.h"
#import  <VMF/VMFramework.h>
#import  <ExternalAccessory/ExternalAccessory.h>

@interface VMFMainViewController : UIViewController <VMFFlipsideViewControllerDelegate,
      UIPopoverControllerDelegate, VFIControlDelegate, VFIPinpadDelegate>
{
    IBOutlet UITextView *tv;
    IBOutlet UILabel *connectedLabel;
}
@property (strong, nonatomic) UIPopoverController *flipsidePopoverController;
-(IBAction) keypadOn:(id)sender;
-(IBAction) keypadOff:(id)sender;
-(IBAction) beepOn:(id)sender;
-(IBAction) beepOff:(id)sender;
-(IBAction) MSROn:(id)sender;
-(IBAction) MSROff:(id)sender;
-(IBAction) creditTest:(id)sender;
-(IBAction) debitTest:(id)sender;
-(IBAction) PINEntryTest:(id)sender;
-(IBAction) obtainTrackData:(id)sender;
-(IBAction) promptUser:(id)sender;
-(IBAction) clear:(id)sender;
@end
```

VMFFlipsideViewController.h:

```
//  VMFFlipsideViewController.h
//  VMF Device Sample
//

#import <UIKit/UIKit.h>
#import  <VMF/VMFramework.h>
#import  <ExternalAccessory/ExternalAccessory.h>

@class VMFFlipsideViewController;

@protocol VMFFlipsideViewControllerDelegate
- (void)flipsideViewControllerDidFinish:(VMFFlipsideViewController *)controller;
@end

@interface VMFFlipsideViewController : UIViewController <VFIBarcodeDelegate> {
    IBOutlet UITextView *tv;
    IBOutlet UILabel *connectedLabel;
}

@property (weak, nonatomic) id <VMFFlipsideViewControllerDelegate> delegate;

- (IBAction)done:(id)sender;
- (IBAction)scannerOn:(id)sender;
- (IBAction)scannerOff:(id)sender;
- (IBAction)scanner1DMode:(id)sender;
- (IBAction)scanner2DMode:(id)sender;
- (IBAction)triggerOn:(id)sender;
- (IBAction)triggerOff:(id)sender;
- (IBAction)setSoft:(id)sender;
- (IBAction)setPassive:(id)sender;
- (IBAction)setEdge:(id)sender;
- (IBAction)setLevel:(id)sender;
- (IBAction)beepOn:(id)sender;
- (IBAction)beepOff:(id)sender;
- (IBAction)clear:(id)sender;
@end
```

5.10.9.8 Step 8: Configure Method File

In the header file, perform the following:

- import the Singleton class VFDevice.h

```
#import "VMFMainViewController.h"
#import "VFDevice.h"
```

- set delegate and initialize barcode, pinpad, and control objects in the `viewDidLoad` methods. Reference: Allocate/initialize VFDevice
- Place these in your MainViewController.m file:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    [[VFDevice pinPad] setDelegate:self];
    [[VFDevice pinPad] initDevice];

    [[VFDevice control] setDelegate:self];
    [[VFDevice control] initDevice];
}
```

- Place these in your VMFFlipsideViewController.m file:

```
- (void)viewDidLoad
{
   [super viewDidLoad];
   [[VFDevice barcode] setDelegate:self];
   [[VFDevice barcode] initDevice];
}
```

- Implement protocol delegate pinpadConnected: (VFIPinpadDelegate-p) to monitor connect/disconnect events and modify our connection label upon change. Reference: Implement optional delegate protocols

```
- (void) pinpadConnected:(BOOL)isConnected{
            if (isConnected) {
            [connectedLabel setText:@"PINPAD CONNECTED"];
            [connectedLabel setTextColor:[UIColor greenColor]];
                        }
            else{
            [connectedLabel setText:@"PINPAD DISCONNECTED"];
            [connectedLabel setTextColor:[UIColor whiteColor]];
        }
}
```

- Implement protocol delegate barcodeScanData:barcodeType: (VFIBarcodeDelegate-p) to receive barcode scan data. Put this in your VMFFlipsideView.m file. Reference: Implement optional delegate protocols

```
- (void) barcodeScanData: (NSData *)data barcodeType:(int)thetype{
    NSString* barcode = [[NSString alloc] initWithData:data
        encoding:NSUTF8StringEncoding];
    tv.text = [NSString stringWithFormat:@"%@\nBarcode Received: %@",tv.text,barcode];
}
```

- Implement protocol delegate pinpadMSRData:expMonth:expYear:track1Data:track2Data: (VFIPinpadDelegate-p) to receive unsolicited card swipe data. Put this in your VMFMainViewController.m file. Reference: Implement optional delegate protocols

```
- (void) pinpadMSRData:(NSString*)pan  expMonth:(NSString*)month
              expYear:(NSString*)year  track1Data:(NSString*)track1
            track2Data:(NSString*)track2{
    tv.text = [NSString stringWithFormat:@"%@\nPAN: %@\nExp: %@/%@\nTrack1: %@\nTrack2: %@",
              tv.text,pan,month,year,track1,track2];
}
```

- Implement the button press methods
- For VMFMainViewController:

```
-(IBAction) creditTest:(id)sender{
 //Credit card transaction demo flow
 [[VFDevice pinPad] obtainTrack2Data:30 optionalAmount:0.00];        // 1. Get card data from user
 //[[VFDevice pinPad] obtainCardData:0 displayIdle:YES];             // (Optional) Use this for retrieving
       manual card entry
```

```
 tv.text = [NSString stringWithFormat:@"Received Track2: %@\n%@",        // 2. Retrieve encrypted track2
       data from VFICardData object
            [[NSString alloc] initWithData:
            [VFDevice pinPad].vfiCardData.track2
             encoding:NSUTF8StringEncoding], tv.text];
 // 3. Pass this data along to your method for building XML
 // 4. Send the XML to the gateway and handle response appropriately
}
-(IBAction) debitTest:(id)sender{
 //Credit card transaction demo flow
 [[VFDevice pinPad] obtainCardData:0 displayIdle:30];                // 1. Get card data from user
 tv.text = [NSString stringWithFormat:@"Acct Num: %@\n%@",               // 2. Retrieve encrypted track2
       data from VFICardData object
             [VFDevice pinPad].vfiCardData.accountNumber,tv.text];

 [[VFDevice pinPad] acceptEncryptPINMessage:[VFDevice pinPad].vfiCardData.accountNumber minPIN:4 maxPIN:8
      requirePIN:YES firstMessage:@"Enter" secondMessage:@"PIN" processingMessage:@"PIN Accepted"];

 tv.text = [NSString stringWithFormat:@"KSN: %@\nPINBlock:%@\n%@", [VFDevice pinPad].vfiEncryptionData.
      serialNumber, [VFDevice pinPad].vfiEncryptionData.pinBlockStr, tv.text];

 // 3. Pass this data along to your method for building XML
 // 4. Send the XML to the gateway and handle response appropriately
}
-(IBAction) beepOn:(id)sender{
 [[VFDevice control] keypadBeepEnabled:YES];
}
-(IBAction) beepOff:(id)sender{
 [[VFDevice control] keypadBeepEnabled:NO];
}
-(IBAction) MSROn:(id)sender{
 [[VFDevice pinPad] enableMSR];
}
-(IBAction) MSROff:(id)sender{
 [[VFDevice pinPad] disableMSR];
}
-(IBAction) keypadOn:(id)sender{
 [[VFDevice control] keypadEnabled:YES];
}
-(IBAction) keypadOff:(id)sender{
 [[VFDevice control] keypadEnabled:NO];
}
-(IBAction) obtainTrackData:(id)sender{
 [[VFDevice pinPad] obtainTrack2Data:30 optionalAmount:0.00];
 tv.text = [NSString stringWithFormat:@"Received Track2: %@\n%@",
            [[NSString alloc] initWithData:
             [VFDevice pinPad].vfiCardData.track2
                                 encoding:NSUTF8StringEncoding], tv.text];
}
-(IBAction) PINEntryTest:(id)sender{
 [[VFDevice pinPad] acceptEncryptPINMessage:@"5454545454545454" minPIN:4 maxPIN:8 requirePIN:YES
      firstMessage:@"Please enter" secondMessage:@"your PIN" processingMessage:@"Thank you"];
 tv.text = [NSString stringWithFormat:@"KSN: %@\nPINBlock:%@\n%@", [VFDevice pinPad].vfiEncryptionData.
      serialNumber, [VFDevice pinPad].vfiEncryptionData.pinBlockStr, tv.text];
}
-(IBAction) promptUser:(id)sender{
 [[VFDevice pinPad] combinedPromptCommand:15.00 language:0 transID:0 menuArray:@"01020304050607" surcharge:
      0.00 accountNum:@"4005550837170001" tipOptions:@"0**" cashBackOptions:@"000" surchargeOptions:1];
 tv.text = [NSString stringWithFormat:@"Tip Amount: %f\nCashBack Amount: %f\nSurcharge Indicator:
       %d\nAmount Confirmed: %d\nAccount Selected: %d\nCashback Selected: %f\nTip Selected: %d\n%@", [VFDevice pinPad].
      vfiDataEntry.tipAmount,[VFDevice pinPad].vfiDataEntry.cashbackAmount,[VFDevice pinPad].vfiDataEntry.
      surchargeIndicator,[VFDevice pinPad].vfiDataEntry.amountConfirmed,[VFDevice pinPad].vfiDataEntry.accountSelected,[
      VFDevice pinPad].vfiDataEntry.cashbackAmount,[VFDevice pinPad].vfiDataEntry.tipSelected,tv.text];
}
-(IBAction) clear:(id)sender{
 tv.text = @"";
}
```

- For VMFFlipsideViewController:

```
-(IBAction) scannerOn:(id)sender{
[[VFDevice barcode] startScan];
}
-(IBAction)scannerOff:(id)sender{
[[VFDevice barcode] abortScan];
}
-(IBAction)scanner1DMode:(id)sender{
[[VFDevice barcode] setScanner1D];
}
```

```
-(IBAction)scanner2DMode:(id)sender{
[[VFDevice barcode] setScanner2D];
}
-(IBAction)triggerOn:(id)sender{
[[VFDevice barcode] sendTriggerEvent:YES];
}
-(IBAction)triggerOff:(id)sender{
[[VFDevice barcode] sendTriggerEvent:NO];
}
-(IBAction)setSoft:(id)sender{
[[VFDevice barcode] setSoft];
}
-(IBAction)setEdge:(id)sender{
[[VFDevice barcode] setEdge];
}
-(IBAction)setLevel:(id)sender{
[[VFDevice barcode] setLevel];
}
-(IBAction)setPassive:(id)sender{
[[VFDevice barcode] setPassive];
}
-(IBAction)beepOn:(id)sender{
[[VFDevice barcode] beepOnParsedScan:YES];
}
-(IBAction)beepOff:(id)sender{
[[VFDevice barcode] beepOnParsedScan:NO];
}
-(IBAction)clear:(id)sender{
tv.text=@"";
}
```

```
//
//  VMFMainViewController.m
//  VMF Device Sample
//
//  Created by James Gnann on 1/10/14.
//  Copyright (c) 2014 VeriFone, Inc. All rights reserved.
//

#import "VMFMainViewController.h"
#import "VMFFlipsideViewController.h"
#import "VFDevice.h"
#import <VMF/VFICardData.h>
#import <VMF/VFIEncryptionData.h>
#import <VMF/VFIDataEntry.h>

@interface VMFMainViewController ()
@end

@implementation VMFMainViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [[VFDevice pinPad] setDelegate:self];
    [[VFDevice pinPad] initDevice];

    [[VFDevice control] setDelegate:self];
    [[VFDevice control] initDevice];
}

-(void) pinpadConnected: (BOOL)isConnected{
    if (isConnected) {
        [connectedLabel setText:@"PINPAD CONNECTED"];
        [connectedLabel setTextColor:[UIColor greenColor]];
    }
    else{
        [connectedLabel setText:@"PINPAD DISCONNECTED"];
        [connectedLabel setTextColor:[UIColor whiteColor]];
    }
}

-(void) pinpadMSRData:(NSString *)pan expMonth:(NSString *)month expYear:(NS
    track1Data:(NSString *)track1 track2Data:(NSString *)track2{
    tv.text = [NSString stringWithFormat:@"PAN: %@\nExp: %@/%@\nTrack1: %@\n
               pan,month,year,track1,track2,tv.text];
}

-(IBAction) creditTest:(id)sender{
    //Credit card transaction demo flow
    [[VFDevice pinPad] obtainTrack2Data:30 optionalAmount:0.00];        // 1
        user
    //[[VFDevice pinPad] obtainCardData:0 displayIdle:YES];             // (
        for retrieving manual card entry

    tv.text = [NSString stringWithFormat:@"Received Track2: %@\n%@",
        encrypted track2 data from VFICardData object
```

```
//
//  VMFFlipsideViewController.m
//  VMF Device Sample
//
//  Created by James Gnann on 1/10/14.
//  Copyright (c) 2014 VeriFone, Inc. All rights reserved.
//

#import "VMFFlipsideViewController.h"
#import "VFDevice.h"

@interface VMFFlipsideViewController ()
@end

@implementation VMFFlipsideViewController
- (void)awakeFromNib
{
    self.preferredContentSize = CGSizeMake(320.0, 480.0);
    [super awakeFromNib];
}
- (void)viewDidLoad
{
    [super viewDidLoad];
    [[VFDevice barcode] setDelegate:self];
    [[VFDevice barcode] initDevice];
}
-(void) barcodeConnected:(BOOL)isConnected{
    if(isConnected){
        connectedLabel.text = @"BARCODE CONNECTED";
        [connectedLabel setTextColor:[UIColor greenColor]];
    }
    else{
        connectedLabel.text = @"BARCODE DISCONNECTED";
        [connectedLabel setTextColor:[UIColor whiteColor]];
    }
}
-(void) barcodeScanData:(NSData *)data barcodeType:(int)thetype{
    NSString* barcode = [[NSString alloc] initWithData:data encoding:NSUTF8St
    tv.text = [NSString stringWithFormat:@"%@\nBarcode Received: %@", tv.text
}
-(IBAction) scannerOn:(id)sender{
    [[VFDevice barcode] startScan];
}
-(IBAction)scannerOff:(id)sender{
    [[VFDevice barcode] abortScan];
}
-(IBAction)scanner1DMode:(id)sender{
    [[VFDevice barcode] setScanner1D];
}
-(IBAction)scanner2DMode:(id)sender{
    [[VFDevice barcode] setScanner2D];
}
-(IBAction)triggerOn:(id)sender{
    [[VFDevice barcode] sendTriggerEvent:YES];
}
```

5.11 VMF Best Practices

This section describes the best practice implementation of the VeriFone Mobile Framework with VeriFone e210/e255/e315 on Apple iOS. Best practice is defined as a method or technique that has consistently shown results superior to those achieved by other means, and this is used as a benchmark. Best practices are used to establish consistent and stable operational characteristics when interfacing with the VeriFone e210/e255/e315.

5.11.1 Best Practices Guidelines

- Establish a Singleton Class to access e210/e255/e315
- Monitor the initialized delegates protocols
- Delegate Protocol Management
- Plan for device resets/modification of settings
- Power down scanner on application exit
- Power down scanner after sending barcode commands
- Let framework manage all connect/disconnect events on background aware app
- Determine proper barcode scanning mode
- VMF Threading
- Keypad Handling
- MSR Reader
- Updating e210/e255/e315 Software

5.11.2 Establish a Singleton Class to access e210/e255/e315:

In software engineering, the singleton pattern is a design pattern that restricts the instantiation of a class to one object. This is useful because we need exactly one object of pinpad, barcode, and control to be accessed throughout the project. Below is example header and method file to establish a Singleton Class named VFDevice:

```
//
//  VFDevice.h
//  Singleton Class
//

#import    <VMF/VMFramework.h>
#import     <ExternalAccessory/ExternalAccessory.h>

@interface VFDevice : NSObject

+ (VFIPinpad *) pinPad;
+ (VFIBarcode *) barcode;
+ (VFIControl *) control;

@end


//
//  VFDevice.m
//  Singleton Class
//
//

#import "VFDevice.h"
```

```
@implementation VFDevice

+ (VFIPinpad *) pinPad{
    static VFIPinpad *_pinPad = nil;
    if (_pinPad == nil) {
        _pinPad = [[VFIPinpad alloc] init];
    }
    return _pinPad;
}
+ (VFIBarcode *) barcode{
    static VFIBarcode *_barcode = nil;
    if (_barcode == nil) {
        _barcode = [[VFIBarcode alloc] init];
    }
    return _barcode;

}
+ (VFIControl *) control{
    static VFIControl *_control = nil;
    if (_control == nil) {
        _control = [[VFIControl alloc] init];
    }
    return _control;
}
@end
```

Once the class is established, you can access it with the VFDevice class level method call. Please reference example provided in Sample Project Tutorial

5.11.3 Monitor the initialized delegates protocols:

Calling the `initDevice` method for any any of the applications on e210/e255/e315 will report back on success/failure of the initialization through the respective delegate protocols:

```
-           (void) pinPadInitialized:(BOOL)isInitialized;
-           (void) controlInitialized:(BOOL)isInitialized;
-           (void) barcodeInitialized:(BOOL)isInitialized;
```

Use these protocols to determine when the applications are initialized and ready to accept commands. An example to initialize barcode settings would be:

```
-(void) barcodeInitialized:(BOOL)isInitialized{
if (isInitialized) {
   [[VFDevice barcode] setScanner2D];
   [[VFDevice barcode] setScanTimeout:5000];
   [[VFDevice barcode] includeAllBarcodeTypes];
   [[VFDevice barcode] barcodeTypeEnabled:TRUE];
  }

}
```

5.11.4 Delegate Protocol Management:

VMF passes information to the delegate specified by the setDelegate method. For example, to set a class as a delegate of the VFIPinpad class, the class you would like to be it's delegate would be set with `[[VFDevice pinpad] setDelegate:self]`. For a single view application, determining what class to make the delegate is straight forward. But for most applications, there are multiple views that need to be able to receive delegate data from VMF.

Best Practices would be to initialize e210/e255/e315 on the main view, and then other views that need access to the e210/e255/e315, you make them delegates of the appropriate VMF class and implement the associated delegate protocols.

Example 1: An application loads to its main screen that requires access to e210/e255/e315 MSR. There is another view that can be navigated to that requires barcode access. In this scenario, the optimal solution would be to init/initDevice of the e210/e255/e315 applications (Pinpad, Barcode, Control) in the main view, set the main view be a delegate

of the VFIPinpad Class and implement the VFIPinpad delegate protocols for the main view, and on the other view requiring barcode, when that view loads, perform a `[[VFDevice barcode] setDelegate:self]` in that view load routines, and make that view a delegate and implement the protocols of the VFIBarcode class.

Example 2: Taking Example 1 a step further, lets say the main view does not require access to e210/e255/e315, but a secondary view requires access to VFIBarcode, and a third view requires access to VFIPinpad. In the main view, you init/initDevice for the e210/e255/e315 (Control, Barcode and Pinpad). The secondary view you would make a delegate of VFIBarcode and upon the view load, set it as a delegate `[[VFDevice barcode] setDelegate:self]`. The third view that requires access to VFIPinpad would be implemented as a delegate of VFIPinpad, and on it's load, it would `[[VFDevice pinpad] setDelegate:self]`.

Note: All delegates are passed data on the Main Thread.

5.11.5 Plan for device resets/modification of settings:

For applications that are background enabled, when the application goes to background, it will disconnect from e210/e255/e315 to allow other applications to take control if needed. When the application returns to the foreground, the framework will automatically perform an `initDevice` for all previously initialized channels. It is important to restore any important device defaults when notification is received that the devices has been successfully initialized (by monitoring the `barcodeInitialized`, `pinpadInitialized`, and `controlInitialized` protocol delegate methods). While the application was in the background, another application may have changed the default settings, a power cycle may have taken place resetting e210/e255/e315 back to defaults, or the e210/e255/e315 may have been swapped with another unit. The application should never assume devices settings have been retained during a backgrounding event. For example, if 2D bar scanning is essential, `[[VFDevice barcode] setScanner2D]` should be executed on each notification of successful device initialization in the barcodeInitialized delegate protocol.

5.11.6 Power down scanner on application exit:

Upon exiting an application, it is best practice to ensure that the power to the barcode engine has been shut down by performing an abortScan (VFIBarcode) on the barcode channel. A common place to do this would be in the `applicationWillTerminate` of the `UIApplication` delegate:

```
-(void)applicationWillTerminate:(UIApplication *)application {
    [[VFDevice barcode] abortScan];
}
```

5.11.7 Power down scanner after sending barcode commands:

Sending any barcode command that executes an ISCP call to the barcode engine will cycle the power to the engine. The power will remain ON after settings have been applied. If the barcode is not going to be used shortly after sending settings to the barcode application, it is recommend that an abortScan (VFIBarcode) is executed after settings have been applied to power down the engine during non-use.

5.11.8 Let framework manage all connect/disconnect events on background aware app

One option is to not allow your application to operate in the background. This is the preferred application model to use with the e210/e255/e315 as management of connection is simplified with connection on application creation, and termination on application exit. To disable background aware application, edit your applications .pList file and set the "Application does not run in background" to "YES".

If your project requires a background aware model, the framework intercepts notifications from the application going to background and returning to foreground, in addition to External Accessory connect/disconnect notifications. Do not attempt to use these notifications to perform additional `initDevice` calls to the applications. After the devices class

has been initialized `[[VFIPinpad alloc] init]`, you only need to perform the first `initDevice`. After that, the framework will disconnect the e210/e255/e315 when going to background, and then reconnect when returning to foreground.

Another option in the .plist introduced with iOS 5 is "Required background modes". Prior to iOS 5, when an application went to the background, it ALWAYS disconnected from the external accessory. With iOS 5, setting Required background modes to "external-accessory" would keep the connection to the device while the app was in the background. VMF overrides this option. In ALL cases, VMF will disconnect the e210/e255/e315 from the application when the application enters background/sleep, and then reconnect automatically upon application returning to foreground.

Supported interface orientation	Array	(4 items)
Application does not run in backgro	Boolean	NO
Required background modes	Array	(1 item)
Item 0	String	
Supported external accessory protoc	Array	(5 items)
Item 0	String	com.verifone.pmr2.control
Item 1	String	com.verifone.pmr2.zontalk

5.11.9 Determine proper barcode scanning mode

There are four scanning modes available:

- **Level Mode**: setLevel (VFIBarcode)- Trigger(s) are connected to the scanning engine. Press and hold the triggers enables scan light. Scanning light will remain on until successful scan, timeout, or trigger released. Scanner light cannot be turned on/off by code.
- **Edge Mode**: setEdge (VFIBarcode)- Trigger(s) are connected to the scanning engine. Press and release the triggers enables scan light. Scanning light will remain on until successful scan or timeout. Scanner light cannot be turned on/off by code.
- **Soft Mode**: setSoft (VFIBarcode)- Trigger(s) are NOT connected to the scanning engine. Pressing the triggers send notification to the application. Scanning can be turned on/off by code
- **Passive Mode**: setPassive (VFIBarcode)- Trigger(s) are NOT connected to the scanning engine. Pressing the triggers will NOT send notification to the application. Scanning can be turned on/off by code

The quickest and most responsive scanning is accomplished by keeping the triggers connected to the scanning engine using Level Mode or Edge Mode. To deactivate triggers while in these modes, but to keep the barcode engine on, setting Passive Mode would be the optimal solution. If you want to intercept trigger presses and control the scanner light via code, then you would use Soft Mode. You would intercept trigger presses with the associated VFIBarcode class delegate protocol:

```
- (void) barcodeTriggerEvent:(int)BCS_TRIGGER {
    if (BCS_TRIGGER == BCS_TRIGGER_RELEASED) {
       [[VFDevice barcode] sendTriggerEvent:FALSE
      } else { [[VFDevice barcode] sendTriggerEvent:TRUE];
     }
  }
```

Please note while Soft Mode offers the most control, it does generate more round trip traffic to the device, and in addition may require additional logic to keep processes in sync as to when to ignore duplicate presses and other similar issues. It is recommended that unless there is a requirement to act up trigger presses other than to activate/deactivate the scanner light (e.g. custom logging statements), scanning should be controlled by the engine receiving the trigger presses with Level Mode and Edge Mode.

Additional information on Soft Mode vs Passive Mode: Soft and Passive more share a single characteristic that Level and Edge mode does not have. While in Soft or Passive mode, you have the ability to control the scanner light/decoding process through an API call: sendTriggerEvent: (VFIBarcode). You cannot control the scanner light/decoding process

while in Level or Edge mode. The sole difference between Soft Mode and Passive Mode is that only Soft mode will receive trigger press notifications through barcodeTriggerEvent: (VFIBarcodeDelegate-p). With Passive Mode, pressing the triggers does not turn on the light/decoding process like with Level/Edge mode, nor does it send a trigger press notification like Soft mode does. Passive mode can be used to process requests for scanning via other notifications in the application, such as a software button on the screen. Passive mode can also be used to deactivate the triggers instead of powering down the scanner in Level or Edge mode.

5.11.10 VMF Threading

Initialization of VMF must be on the main thread. Most of the VMF commands are blocked while waiting for a response. If you desire to bypass blocking, you can execute the commands using performSelectorOnMainThread. While this doesnt offload the processing to separate thread, it does pass the code path to process on a parallel Main thread so your application can continue to execute its current code path.

```
self performSelectorOnMainThread:@selector(startTheBackgroundJob) withObject:nil waitUntilDone:NO];

- (void)startTheBackgroundJob {
[[VFDevice pinPad] getCardData:120 language:0 amount:5.00 otherAmount:0];
}
```

5.11.11 Keypad Handling

By default, when the iOS device is connected to the e210/e255/e315, the keypad will be disabled. This prevents accidental keypad activation (and keypad beeping) while the iOS side of the application pair is being used. As a result, your application must ensure that the keypad is enabled prior to issuing commands to the e210/e255/e315 to accept input (e.g. PIN Entry Requests). This is done with the control application. In addition, you can configure if the keypad beep is silent or not:

```
[[VFDevice control] keypadBeepEnabled:YES] ;
[[VFDevice control] keypadEnabled:NO];
```

5.11.12 MSR Reader

When the e210/e255/e315 is powered up, the MSR Reader will be off by default. The MSR reader must be turned on if your application plans on reading any magnetic stripe card data. If your application is background aware, this event should happen in the pinpadInitialized delegate to ensure the MSR wasnt disabled while the application was in the background:

```
-(void) pinPadInitialized:(BOOL)isInitialized{
   if (isInitialized) {
      [[VFDevice pinPad] enableMSRDualTrack];
   }
}
```

XPI returns track data through two means. The first is capturing a card swipe without a specific request to do so from VMF. This would be considered an unsolicited card swipe and is activated by enableMSR (VFIPinpad) or enableMSR-DualTrack (VFIPinpad).

Unsolicited track data is returned by XPI and passed unaltered/unparsed by VMF to the delegate protocol pinpadMSR-Data:expMonth:expYear:track1Data:track2Data: (VFIPinpadDelegate-p). Any parsing of track data, for instance pulling out customer name from Track1, or service code from Track 2, must be performed by the integrator.

The second method of capturing card information is with getCardData:language:amount:otherAmount: (VFIPinpad), obtainCardData: (VFIPinpad), obtainTrack2Data:optionalAmount: (VFIPinpad). This would be considered a solicited card request.

A solicited card request will "block" code execution until the e210/e255/e315 returns data. This data could be card data, timeout response, or cancel response. This card request will accept card input from all the available methods on the e210/e255/e315. This would include MSR, Chip (card insert), and Contactless (tap) if CTLS software is loaded on e210/e255/e315. Card data is again returned by XPI unaltered by VMF and populated in the VFIPinpad::vfiCardData object. XPI does parse out more information from a solicited card request than an unsolicited card swipe. The possible information that can by passed from XPI in VFICardData() object is:

```
@interface VFICardData : NSObject {
            NSString *AID;
            NSString *appPreferredName;
            NSString *appLabel;
            int serviceCode;
            NSString *cardHolderName;
            NSData *track2;
            NSData *track1;
            NSData *track3;
            NSString *accountNumber;
            NSString *expiryDate;
            NSMutableDictionary *emvTags;
            BOOL isCTLS;
            int entryType;

}
```

5.11.13 Updating e210/e255/e315 Software

Software updates can be performed using VMF if XPI is already running on the e210/e255/e315. If XPI is NOT or already installed on e210/e255/e315, then software can only be installed using a terminal, VeriFone service board, and Verifone cable. Contact your VeriFone representative if this case is applicable.

Software updates through VMF requires a specially signed and packaged .zip archive. This archive contains the files and certificates to be downloaded and a download procedure file. VMF has two methods to pass update files and begin the update process. This first is if a valid network URL is provided that contains location of the .zip archive. The url would be passed to the method updateFromUrl: (VFIPinpad). The second method is you read in the .zip file into a NSData object, and pass that to the framework via updateFromZip: (VFIPinpad);

- Make sure the e210/e255/e315 battery is full or on external power before performing update
- Do not interrupt the update process by shutting down the unit
- Do not attempt to send any files of unknown origin. Only VeriFone certified update packages may be use. Downloading other files may render the e210/e255/e315 inoperable
- Updates over bluetooth take a longer time than non-bluetooth e210/e255/e315. When on Bluetooth, please make sure to stay in range for the duration of the update.
- The e210/e255/e315 will automatically reboot upon completion.

5.12 Secure Messages for US Commands

The following Secure Messages for US Commands Table is used with packets Z2 - displayMessage:clearScreen: (VFIPinpad) or Z3 - displayMessages:message1:message2:clearScreen: (VFIPinpad).

English Prompt	French Prompt	Spanish Prompt
"ID"	"ID"	"ID"

"NO"	"NO"	"NO"
"PO"	"PO"	"PO"
"AMT"	"AMT"	"AMT"
"EMP"	"EMP"	"EMP"
"GOV"	"GOV"	"GOB"
"JOB"	"JOB"	"TRAB"
"OTH"	"OTH"	"OTR"
"ZIP"	"ZIP"	"CP"
"CKSV"	"CKSV"	"CKSV"
"CODE"	"CODE"	"CODIGO"
"DATA"	"DATA"	"INFO"
"DATE"	"DATE"	"FECHA"
"DEPT"	"DEPT"	"DEPTO"
"IDNO"	"IDNO"	"NUMID"
"PONO"	"PONO"	"NUMPO"
"USER"	"USER"	"USUARIO"
"BADGE"	"BADGE"	"DISTINTIVO"
"CLOCK"	"CLOCK"	"RELOJ"
"DIVNO"	"DIVNO"	"NUMDIV"
"EMPNO"	"EMPNO"	"NUMEMP"
"ENTER"	"ENTER"	"INGRESE"
"EQUIP"	"EQUIP"	"EQUIP"
"JOBNO"	"JOBNO"	"NUMTRAB"
"MAXWD"	"MAXWD"	"MAXWD"
"MOSYN"	"MOSYN"	"MOSYN"
"AHORRO"	"AHORRO"	"AHORRO"
"AMOUNT"	"AMOUNT"	"MONTO"
"CKSVCC"	"CKSVCC"	"CKSVCC"
"CLIENT"	"CLIENT"	"CLIENTE"
"DEPTNO"	"DEPTNO"	"NUMDEPTO"
"DRIVER"	"DRIVER"	"CONDUCTOR"
"NUMBER"	"NUMBER"	"NUMERO"
"OTHOTH"	"OTHOTH"	"OTROTR"
"POSTAL"	"POSTAL"	"POSTAL"
"SERIAL"	"SERIAL"	"SERIAL"
"SOCSEC"	"SOCSEC"	"SEGSOC"
"USERID"	"USERID"	"IDUSUAR"
"VEHTAG"	"VEHTAG"	"TAGVEH"
"CLOCKNO"	"CLOCKNO"	"NUMRELOJ"
"CONTROL"	"CONTROL"	"CONTROL"
"DEPOSIT"	"DEPOSIT"	"DEPOSITO"
"ENTERID"	"ENTERID"	"INGRESEID"
"ENTERRC"	"ENTERRC"	"INGRESERC"
"ENTERSS"	"ENTERSS"	"INGRESESS"

"EQUIPCD"	"EQUIPCD"	"EQUIPCD"
"FINANCE"	"FINANCE"	"FINANCIA"
"OKYESNO"	"OKYESNO"	"OKSINO"
"PAYROLL"	"PAYROLL"	"PLANSUELDO"
"READING"	"READING"	"LEYENDO"
"REENTER"	"REENTER"	"REINGRESE"
"SAVINGS"	"SAVINGS"	"AHORROS"
"VEHICLE"	"VEHICLE"	"VEHICULO"
"WORKORD"	"WORKORD"	"ORDTRAB"
"CASHBACK"	"CASHBACK"	"REINTEGOR"
"CHECKING"	"CHECKING"	"CHECKEANDO"
"CUSTOMER"	"CUSTOMER"	"CLIENTE"
"DATEMMYY"	"DATEMMYY"	"FECHAMMAA"
"DIGITZIP"	"DIGITZIP"	"DIGITCP"
"DIVISION"	"DIVISION"	"DIVISION"
"DRIVERID"	"DRIVERID"	"IDCONDUCTOR"
"ENTEREMP"	"ENTEREMP"	"INGRESEEMP"
"ENTERHUB"	"ENTERHUB"	"INGRESEHUB"
"ENTERJOB"	"ENTERJOB"	"INGRESETRAB"
"ENTERLIC"	"ENTERLIC"	"INGRESELIC"
"ENTERREF"	"ENTERREF"	"INGRESEREF"
"ENTERTAG"	"ENTERTAG"	"INGRESETAG"
"GOVBADGE"	"GOVBADGE"	"INSIGNIAGOB"
"IDNUMBER"	"IDNUMBER"	"NUMEROID"
"MMDDYYYY"	"MMDDYYYY"	"MMDDAAAA"
"ODOMETER"	"ODOMETER"	"ODOMETRO"
"PONUMBER"	"PONUMBER"	"NUMEROPO"
"SECURITY"	"SECURITY"	"SEGURIDAD"
"WDINQXFR"	"WDINQXFR"	"WDINQXFR"
"CORRIENTE"	"CORRIENTE"	"CORRIENTE"
"DRIVERSID"	"DRIVERSID"	"IDCONDUCTORES"
"ENTERAUTH"	"ENTERAUTH"	"INGRESEAUT"
"ENTERCARD"	"ENTERCARD"	"INGRESETARJETA"
"ENTERCLUB"	"ENTERCLUB"	"INGRESECLUB"
"ENTERCODE"	"ENTERCODE"	"INGRESECODIGO"
"ENTERDATA"	"ENTERDATA"	"INGRESEINFO"
"ENTERDEPT"	"ENTERDEPT"	"INGRESEDEPTO"
"ENTERDRVR"	"ENTERDRVR"	"INGRESECONDUCTOR"
"ENTERFOOD"	"ENTERFOOD"	"INGRESECOMIDA"
"ENTERMENU"	"ENTERMENU"	"INGRESEMENU"
"ENTERYOUR"	"ENTERYOUR"	"INGRESESU"
"EQUIPCODE"	"EQUIPCODE"	"CODIGOEQUIP"
"JOBNUMBER"	"JOBNUMBER"	"NUMTRABAJO"
"OTHOTHOTH"	"OTHOTHOTH"	"OTROTROTR"

"TERMFEEOF"	"TERMFEEOF"	"TASATERMINAL"
"VEHICLEID"	"VEHICLEID"	"IDVEHICULO"
"WORKORDER"	"WORKORDER"	"ORDENTRABAJO"
"CUSTOMERID"	"CUSTOMERID"	"IDCLIENTE"
"ENTERFLEET"	"ENTERFLEET"	"INGRESEFLOTA"
"ENTERJOBID"	"ENTERJOBID"	"INGRESEIDTRABAJO"
"ENTERJOBNO"	"ENTERJOBNO"	"INGRESENUMTRABAJO"
"ENTERPHONE"	"ENTERPHONE"	"INGRESETELEFONO"
"ENTERROUTE"	"ENTERROUTE"	"INGRESERUTA"
"ENTERSTORE"	"ENTERSTORE"	"INGRESETIENDA"
"ENTERVALUE"	"ENTERVALUE"	"INGRESEVALOR"
"ENTREDATOS"	"ENTREDATOS"	"INGRESEDATOS"
"ENTREDEPTO"	"ENTREDEPTO"	"INGRESEDEPTO"
"OVERMAXAMT"	"OVERMAXAMT"	"SOBRELIMMAX"
"PLEASEWAIT"	"PLEASEWAIT"	"PORFAVORESPERE"
"PROCESSING"	"PROCESSING"	"PROCESANDO"
"REENTERVEH"	"REENTERVEH"	"REINGRESEVEH"
"CHECKSAVING"	"CHECKSAVING"	"CHECKAHORRO"
"CREDITCHECK"	"CREDITCHECK"	"CHECKCREDITO"
"ENTERAMOUNT"	"ENTERAMOUNT"	"INGRESEMONTO"
"ENTERCARDID"	"ENTERCARDID"	"INGRESEIDTARJETA"
"ENTERCLIENT"	"ENTERCLIENT"	"INGRESECLIENTE"
"ENTERCUSTID"	"ENTERCUSTID"	"INGRESEIDCLIENTE"
"ENTERDEPTNO"	"ENTERDEPTNO"	"INGRESENUMDEPTO"
"ENTERDRIVER"	"ENTERDRIVER"	"INGRESECONDUCTOR"
"ENTERKEYFOB"	"ENTERKEYFOB"	"INGRESELLAVEFOB"
"ENTERMEMBER"	"ENTERMEMBER"	"INGRESEMIEMBRO"
"ENTERPOINTS"	"ENTERPOINTS"	"INGRESEPUNTOS"
"ENTERSOCSEC"	"ENTERSOCSEC"	"INGRESESEGSOC"
"ENTERSTREET"	"ENTERSTREET"	"INGRESECALLE"
"ENTERUSERID"	"ENTERUSERID"	"INGRESEIDUSUARIO"
"FLEETNUMBER"	"FLEETNUMBER"	"NUMEROFLOTA"
"PLEASEENTER"	"PLEASEENTER"	"PORFAVORINGRESE"
"REENTERYOUR"	"REENTERYOUR"	"REINGRESESU"
"ROUTENUMBER"	"ROUTENUMBER"	"NUMERORUTA"
"SAVINGCHECK"	"SAVINGCHECK"	"CHECKAHORRO"
"TOINPUTDATA"	"TOINPUTDATA"	"PARAINGRESARINFO"
"WILLBEADDED"	"WILLBEADDED"	"SERAAGREGADO"
"CUSTOMERCODE"	"CUSTOMERCODE"	"CODIGOCLIENTE"
"DRIVERNUMBER"	"DRIVERNUMBER"	"NUMEROCONDUCTOR"
"ENTERACCOUNT"	"ENTERACCOUNT"	"INGRESECUENTA"
"ENTERCARWASH"	"ENTERCARWASH"	"INGRESELAVADO"
"ENTERCONTROL"	"ENTERCONTROL"	"INGRESECONTROL"
"ENTERCUSTREF"	"ENTERCUSTREF"	"INGRESEREFCLI"

"ENTERFLEETNO"	"ENTERFLEETNO"	"INGRESENUMFLOTA"
"ENTERLICENSE"	"ENTERLICENSE"	"INGRESELICENCIA"
"ENTERLOYALTY"	"ENTERLOYALTY"	"INGRESEFIDELIDAD"
"ENTERMESSAGE"	"ENTERMESSAGE"	"INGRESEMENSAJE"
"ENTERPRODUCT"	"ENTERPRODUCT"	"INGRESEPRODUCTO"
"ENTERSECCODE"	"ENTERSECCODE"	"INGRESECODSEC"
"ENTERSERVICE"	"ENTERSERVICE"	"INGRESESERVICIO"
"ENTERTRAILER"	"ENTERTRAILER"	"INGRESETRAILER"
"ENTERVEHICLE"	"ENTERVEHICLE"	"INGRESEVEHICULO"
"ENTERZIPCODE"	"ENTERZIPCODE"	"INGRESECODPOSTAL"
"ORFOROFFLINE"	"ORFOROFFLINE"	"OPARAFUERADELINEA"
"PLSSLIDECARD"	"PLSSLIDECARD"	"FVRDESLICETRAJETA"
"REENTERCNTRL"	"REENTERCNTRL"	"REINGRESECNTRL"
"SECURITYCODE"	"SECURITYCODE"	"CODSEGURIDAD"
"DRIVERLICENSE"	"DRIVERLICENSE"	"LICENCIACONDUCTOR"
"ENTERCASHBACK"	"ENTERCASHBACK"	"INGRESEREINTEGRO"
"ENTERCHARGETO"	"ENTERCHARGETO"	"INGRESECARGARA"
"ENTERCUSTCODE"	"ENTERCUSTCODE"	"INGRESECODCLIENTE"
"ENTERCUSTDATA"	"ENTERCUSTDATA"	"INGRESEINFOCLIENTE"
"ENTERCUSTOMER"	"ENTERCUSTOMER"	"INGRESENUMCLIENTE"
"ENTERDIGITZIP"	"ENTERDIGITZIP"	"INGRESEDIGCP"
"ENTERDRIVERID"	"ENTERDRIVERID"	"INGRESEIDCONDUCTOR"
"ENTEREMPLOYEE"	"ENTEREMPLOYEE"	"INGRESEEMPLEADO"
"ENTERFLEETPIN"	"ENTERFLEETPIN"	"INGRESEPINFLOTA"
"ENTERFREQUENT"	"ENTERFREQUENT"	"INGRESEFRECUENTE"
"ENTERIDNUMBER"	"ENTERIDNUMBER"	"INGRESENUMEROID"
"ENTERLOCATION"	"ENTERLOCATION"	"INGRESEUBICACION"
"ENTERODOMETER"	"ENTERODOMETER"	"INGRESEODOMETRO"
"ENTERPERSONAL"	"ENTERPERSONAL"	"INGRESEPERSONAL"
"ENTERUSERDATA"	"ENTERUSERDATA"	"INGRESEINFOUSUARIO"
"ENTERVERIPASS"	"ENTERVERIPASS"	"INGRESEVERIPASS"
"ENTREIDCHOFER"	"ENTREIDCHOFER"	"INGRESEIDCHOFER"
"ENTRENUMVEHIC"	"ENTRENUMVEHIC"	"INGRESENUMVEHIC"
"ENTREODOMETRO"	"ENTREODOMETRO"	"INGRESEODOMETRO"
"GOVERNMTBADGE"	"GOVERNMTBADGE"	"PLACAGOBIERNO"
"INVALIDCHOICE"	"INVALIDCHOICE"	"OPCIONINVALIDA"
"PLEASEREENTER"	"PLEASEREENTER"	"PORFAVORREINGRESE"
"POSTALFINANCE"	"POSTALFINANCE"	"FINANCIACIONPOSTAL"
"REENTERDRIVID"	"REENTERDRIVID"	"REINGRESEIDCONDUCTOR"
"SELECTACCOUNT"	"SELECTACCOUNT"	"SELECCCUENTA"
"VEHICLENUMBER"	"VEHICLENUMBER"	"NUMEROVEHICULO"
"WITHDRAWALAMT"	"WITHDRAWALAMT"	"MONTORETIRO"
"ENTERAUTHNUMBR"	"ENTERAUTHNUMBR"	"INGRESENUMAUT"
"ENTERBIRTHDATE"	"ENTERBIRTHDATE"	"INGRESEFECHNAC"

"ENTERDOBMMDDYY"	"ENTERDOBMMDDYY"	"INGRESEFDNMMDDAA"
"ENTERDRIVERLIC"	"ENTERDRIVERLIC"	"INGRESELICCONDUCTOR"
"ENTEREXPIRDATE"	"ENTEREXPIRDATE"	"INGRESEFECHAEXP"
"ENTERFLEETDATA"	"ENTERFLEETDATA"	"INGRESEINFOFLOTA"
"ENTERHUBNUMBER"	"ENTERHUBNUMBER"	"INGRESENUMEROHUB"
"ENTERHUBOMETER"	"ENTERHUBOMETER"	"INGRESEMEDIDORHUB"
"ENTERREFERENCE"	"ENTERREFERENCE"	"INGRESEREFERENCIA"
"ENTERTRAILERID"	"ENTERTRAILERID"	"INGRESEIDTRAILER"
"ENTERTRAILERNO"	"ENTERTRAILERNO"	"INGRESENUMTRAILER"
"ENTERVEHICLEID"	"ENTERVEHICLEID"	"INGRESEIDVEHICULO"
"ENTREIDUSUARIO"	"ENTREIDUSUARIO"	"INGRESEIDUSUARIO"
"ODOMETERNUMBER"	"ODOMETERNUMBER"	"NUMEROODOMETRO"
"PREVIOUSSTREET"	"PREVIOUSSTREET"	"CALLEPREVIA"
"REENTERCONTROL"	"REENTERCONTROL"	"REINGRESECONTROL"
"REENTERSECCODE"	"REENTERSECCODE"	"REINGRESECODSEC"
"REENTERVEHICLE"	"REENTERVEHICLE"	"REINGRESEVEHICULO"
"REENTERZIPCODE"	"REENTERZIPCODE"	"INGRESECODPOSTAL"
"TRANSFERAMOUNT"	"TRANSFERAMOUNT"	"MONTOTRANSFER"
"TYPEOFTRANSFER"	"TYPEOFTRANSFER"	"TIPODETRANSFER"
"WAITINGFOREMPL"	"WAITINGFOREMPL"	"ESPERANDOPOREMPL"
"ENTERCUSTOMERID"	"ENTERCUSTOMERID"	"INGRESEIDCLIENTE"
"ENTEREXPIRATION"	"ENTEREXPIRATION"	"INGRESEEXPIRACION"
"ENTERMEMBERSHIP"	"ENTERMEMBERSHIP"	"INGRESEMEMBRESIA"
"ODOMETERREADING"	"ODOMETERREADING"	"LECTURAODOMETRO"
"PREVIOUSZIPCODE"	"PREVIOUSZIPCODE"	"CODPOSTPREVIO"
"REENTERDRIVERID"	"REENTERDRIVERID"	"REINGRESEIDCONDUCTO R"
"REENTERODOMETER"	"REENTERODOMETER"	"REINGRESEODOMETRO"
"RESTRICTIONCODE"	"RESTRICTIONCODE"	"CODRESTRICCION"
"ENTERFLEETNUMBER"	"ENTERFLEETNUMBER"	"INGRESENUMEROFLOTA"
"ENTERROUTENUMBER"	"ENTERROUTENUMBER"	"INGRESENUMERORUTA"
"PROCESSINGPINPAD"	"PROCESSINGPINPAD"	"PROCESANDOTECLADO"
"REMAININGBALANCE"	"REMAININGBALANCE"	"SALDODISPONIBLE"
"ENTERCUSTOMERCODE"	"ENTERCUSTOMERCODE"	"INGRESECODIGOCLIENTE"
"ENTERCUSTOMERDATA"	"ENTERCUSTOMERDATA"	"INGRESEINFOCLIENTE"
"ENTERDRIVERNUMBER"	"ENTERDRIVERNUMBER"	"INGRESENUMEROCONDU CTOR"
"ENTERSECURITYCODE"	"ENTERSECURITYCODE"	"INGRESECODIGOSEGURID AD"
"ENTERDRIVERLICENSE"	"ENTERDRIVERLICENSE"	"INGRESELICENCIAVEHICU LO"
"ENTERVEHICLENUMBER"	"ENTERVEHICLENUMBER"	"INGRESENUMEROVEHICU LO"
"ENTEREXPIRATIONDATE"	"ENTEREXPIRATIONDATE"	"INGRESEFECHAEXPIRACI ON"
"ENTERODOMETERREADING"	"ENTERODOMETERREADING"	"INGRESELECTURAODOME TRO"
"ENTERRESTRICTIONCODE "	"ENTERRESTRICTIONCODE "	"INGRESECODIGORESTRICCIO-N"

"CREDIT"	"CREDIT"	"CREDITO"
"DEBIT"	"DEBIT"	"DEBITO"
ENTERPIN	"ENTERPIN"	"INGRESEPIN"
ENTERVCODE	"ENTERVCODE"	INGRESECODIGOV
"LASTFOURDIGITS"	"LASTFOURDIGITS"	ULTIMOS CUATRO DIGITOS
LASTDIGITS	LASTDIGITS	ULTDIGITOS
INVALIDCARD	CARTEINVALIDE	TARJETANOVALIDO"
DECLINED	REFUSE	RECHAZADA
PLEASESWIPECARD	SVPPASSERCARTE	PORFAVORDESLIZETARJE TA
"TIP"	"TIP"	"PROPINA"
"TIPPERCENT"	"TIPPERCENT"	"PORCIENTODEPROPINA"
"CASHBACK"	"CASHBACK"	"DEVOLVERDINERO"
"SURCHARGEFEE"	"SURCHARGEFEE"	"RECARGODEPAGO"

5.13 PINPad Parameters

Parameter	Description	Values
ENTRYMODE	Determines whether the device will allow manual account number/ expiration date entry, swipe data entry, or both.	0 - (Default) Both manual entry and card swipe supported 1 - Manual entry only is supported 2 - Swipe only is supported
PROMPTO	Timeout value for Prompt	Numeric (0-99)
DIGITO	Timeout value for Inter-digit PIN Entry	Numeric (0-99)
BALTO	Timeout value for Display Balance	Numeric (0-99)
CBMODE	Turns on/off cash back prompting	Flag (0/1)
CHQ	Sets the checking account value	Alphanumeric
SAV	Sets the saving account value	Alphanumeric
EMVDS	Determines if C34 command displays approved/declined if there is no optional show status parameter sent in the C34 command.	Flag (0/1) 0 - Display 1 - Do not display
CBAMT1	Cash back amount 1	Numeric (0-9999)
CBAMT2	Cash back amount 2	Numeric (0-9999)
CBAMT3	Cash back amount 3	Numeric (0-9999)
CBAMT4	Cash back amount 4	Numeric (0-9999)
EMVLANG	Sets the EMV default language	Flag (0/1)
DEFLANG	Sets the PIN pad default language	Alpha (E/F)
SNREMOVE	Sets the number of characters to be removed from the beginning of the 9-digit device serial number	Numeric (0-9)
SNPREFIX	Sets the prefix to be added in front of the device serial number	Alphanumeric (3 characters)
BACKLITE	Turns on/off the backlight option	Flag (0/1)
VISATTQ	Sets the VISA TTQ	Hexadecimal (8 characters)

OPTFLAG	Sets the Visa Opt-Out Option	Numeric (16 digits)
BAUDRATE	Sets the default Baud Rate	300/1200/2400/9600/115200
FORMAT	Sets the parity setting	8N1/7E1
COMMPORT	Sets the default Comm Port.	2 - RS232 6 - USB using the base U - USB I - Ethernet
URLHOST	Sets URL of the host.	Numeric. 15 Ex: 123.456.789.123
PORT	Host (ECR) Port ID used in IP. Leave empty if do not want to check against Host Port ID.	Numeric. 4
WELCMSG	Sets the IDLE prompt message	Alphanumeric (16 characters)
COUNTRY	Sets the country	Alphanumeric (CANADA/US)
PELAGE	Enable/Disable the check for the Interac AID in the candidate list and the Visa DEBIT application label if tag 5F56 does not exists	Flag (0/1)
TERMPW	VMAC Entry Password	Alphanumeric (6 characters)
LANGMODE	Sets the language mode to unilingual or bilingual	Numeric 1 - unilingual 0 or not set - bilingual
APPCONFRM	Sets an additional application confirmation prompt for EMV transactions when application confirmation is required by the card.	0 - Do not prompt 1 - Prompt if card requires prompting.
XPILANGCNT	Number of languages supported	Numeric. 2-5. Default is 2.
XPILANG0	First default language	Alpha. Default is ENGL.
XPILANG1	Second default language	Alpha. Default is FREN.
XPILANG2	ISO Code for third default language	Alpha.
XPILANG3	ISO Code for fourth default language	Alpha.
XPILANG4	ISO Code for fifth default language	Alpha.
XPIDUKPTIDX	Identifies the DUKPT engine used by XPI	Numeric: 0, 1, or 2 (via packet 19) Default is 0.
XPIENCRPT	Identifies the encryption used	Alpha: MS1DES, MS3DES, DUKPT Default is DUKPT
XPICURR	Identifies the currency code used. Three ways of entering currency code: enter 3-character code, like USD; currency symbol, like $; and backslash + ASCII code, like \190	Alpha. Default is $
XPIPANCHK	Enable/disables PAN checking	Numeric. 0, 1. Value of 0 (default) means no checking, while value of 1 means platform checking

ICCERR	Sets number of times an inserted card failure occurs before returning the error back to the host device	Numeric

5.14 EMV Data Formats

Any references in this document related to the passing of EMV Tag data to and from the processor and the Vx 600BT device running XPI will be in TLV (Tag, Length, Value) Format. The TLV format contains hexadecimal binary data that represents the EMV Tag, the length of the tag data and the associated data.

In an EMV Online Complete transaction, for example, the processor (host) would send any relevant EMV tag information in TLV format which will be provided through the VeriFone Mobile Framework command, completeOnlineEMV:displayResult:hostTags: and passed through to XPI through the interface between the Windows device and the Vx 600BT device. A sample of the TLV data is as follows:

```
<TLV From Host>
8A 02 30 30 91 10 31 31 31 31 31 31 31 31
31 31 31 31 31 31 31 31
```

The tag data in the above sample includes TAG '8A' with a 2 byte data value of 3030; and TAG 91 with a 16 byte data value of '31313131313131313131313131313131'.

The sample below represents an exchange of commands C30 and a C31 response from XPI that illustrates TLV data noted by the green highlighting in the sample packet.

5.14.1 Master Card Example with Request for Tag Information C36

Sample C30 Request

```
02  43  33  30  00  18  C1  01  25  C1  01  01  C1  03  09  11  ~C30~~~~%~~~~~~~
20  C1  03  14  00  05  C1  01  00  C1  01  31  C1  00  03  A6   ~~~~~~~~~~1~~~~
```

Sample C31 Response

```
02  43  33  31  30  30  00  70  4F  07  A0  00  00  00  04  10  ~C3100~pO~~~~~~~
10  9F  12  00  50  0A  4D  61  73  74  65  72  43  61  72  64  ~~~~P~MasterCard
5F  30  02  02  01  5F  20  2A  2A  2A  2A  2A  2A  2A  2A  2A  _0~~~_ ~*********
2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  ****************
2A  2A  57  13  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  2A  ***~************
2A  2A  2A  2A  2A  2A  2A  5A  08  2A  2A  2A  2A  2A  2A  2A  *******Z~*******
2A  5F  24  03  2A  2A  2A  5F  34  01  00  5F  25  03  09  10  *_$~***_4~~_%~~~
31  9F  39  01  05  C2  01  2A  03  BE                          1~9~~~~*~~
```

In the above sample, there are 5 EMV Tags represented in the C31 Response. Those tags are:

Tag	Length	Value
9F12	0 bytes	<NULL>
5F30	2 bytes	0201
57	19 bytes	2A2A2A2A2A2A2A2A2A2A2A2A2-A2A2A2A2A2A2A
5F24	3 bytes	2A2A2A
5F25	3 bytes	091031

All TLV noted EMV Tag information will be handled using this format. In certain noted framework commands, paired arrays with tag names and tag values can be utilized to avoid the need to construct a TLV string to pass through the framework. The framework will handle the appropriate command construction for interacting with the XPI application on the PIN pad.

5.15 General Message Table

Secure messages to be used with General Prompts commands

Msg Id	English Prompt	French Prompt	Spanish Prompt
1	"ENTER"	"ENTREZ"	"INGRESE"
2	"REENTER"	"RE-ENTREZ"	"REINGRESE"
3	"ENTER YOUR"	"ENTREZ VOTRE"	"INGRESE SU"
4	"REENTER YOUR"	"RE-ENTREZ VOTRE"	"REINGRESE SU"
5	"PLEASE ENTER"	"SVP ENTREZ"	"POR FAVOR INGRESE"
6	"PLEASE REENTER"	"SVP RE-ENTREZ"	"POR FAVO REINGRESE"
7	"PO NUMBER"	"No COMMANDE"	"NUMERO PO"
8	"DRIVER ID"	"ID CONDUCTEUR"	"LICENCIA"
9	"ODOMETER"	"ODOMETRE"	"ODOMETRO"
10	"ID NUMBER"	"No IDENT"	"NUMERO ID"
11	"EQUIP CODE"	"CODE EQUIPEMENT"	"CODIGO EQUIP"
12	"DRIVERS ID"	"ID CONDUCTEUR"	"ID CONDUCTOR"
13	"JOB NUMBER"	"No TRAVAIL"	"NUMERO EMP"
14	"WORK ORDER"	"FICHE TRAVAIL"	"ORDEN TRABAJO"
15	"VEHICLE ID"	"ID VEHICULE"	"ID VEHICULO"
16	"ENTER DRIVER"	"ENTR CONDUCTEUR"	"INGRESE CONDUCTOR"
17	"ENTER DEPT"	"ENTR DEPARTEMNT"	"INGRESE DEPT"
18	"ENTER PHONE"	"ENTR No TELEPH"	"INGRESE TELEFONO"
19	"ENTER ROUTE"	"ENTREZ ROUTE"	"INGRESE RUTA"
20	"ENTER FLEET"	"ENTREZ PARC AUTO"	"INGRESE FLOTA"
21	"ENTER JOB ID"	"ENTR ID TRAVAIL"	"INGRESE ID TRABAJO"
22	"ROUTE NUMBER"	"No ROUTE"	"RUTA NUMERO"
23	"ENTER USER ID"	"ID UTILISATEUR"	"INGRESE ID USUARIO"
24	"FLEET NUMBER"	"No PARC AUTO"	"FLOTA NUMERO"
25	"ENTER PRODUCT"	"ENTREZ PRODUIT"	"INGRESE PRODUCTO"
26	"DRIVER NUMBER"	"No CONDUCTEUR"	"CONDUCTOR NUMERO"
27	"ENTER LICENSE"	"ENTREZ PERMIS"	"INGRESE LICENCIA"
28	"ENTER FLEET NO"	"ENT No PARC AUTO"	"INGRESE NRO FLOTA"
29	"ENTER CAR WASH"	"ENTREZ LAVE-AUTO"	"INGRESE LAVADO"
30	"ENTER VEHICLE"	"ENTREZ VEHICULE"	"INGRESE VEHICULO"
31	"ENTER TRAILER"	"ENTREZ REMORQUE"	"INGRESE TRAILER"
32	"ENTER ODOMETER"	"ENTREZ ODOMETRE"	"INGRESE ODOMETRO"
33	"DRIVER LICENSE"	"PERMIS CONDUIRE"	"LICENCIA CONDUCTOR"
34	"ENTER CUSTOMER"	"ENTREZ CLIENT"	"INGRESE CLIENTE"
35	"VEHICLE NUMBER"	"No VEHICULE"	"VEHICULO NUMERO"
36	"ENTER CUST DATA"	"INFO CLIENT"	"INGRESE INFO CLIENTE"
37	"REENTER DRIVID"	"RE-ENTR ID COND"	"REINGRESE ID CHOFER"

38	"ENTER USER DATA"	"INFO UTILISATEUR"	"INGRESE INFO USUARIO"
39	"ENTER CUST CODE"	"ENTR CODE CLIENT"	"INGRESE COD. CLIENTE"
40	"ENTER EMPLOYEE"	"ENTREZ EMPLOYE"	"INGRESE EMPLEADO"
41	"ENTER ID NUMBER"	"ENTREZ No ID"	"INGRESE NUMERO ID"
42	"ENTER DRIVER ID"	"No CONDUCTEUR"	"INGRESE ID CONDUCTOR"
43	"ENTER FLEET PIN"	"NIP PARC AUTO"	"INGRESE PIN DE FLOTA"
44	"ODOMETER NUMBER"	"No ODOMETRE"	"ODOMETRO NUMERO"
45	"ENTER DRIVER LIC"	"PERMIS CONDUIRE"	"INGRESE LIC CONDUCTOR"
46	"ENTER TRAILER NO"	"ENT No REMORQUE"	"INGRESE NRO TRAILER"
47	"REENTER VEHICLE"	"RE-ENTR VEHICULE"	"REINGRESE VEHICULO"
48	"ENTER VEHICLE ID"	"ENTR ID VEHICULE"	"INGRESE ID VEHICULO"
49	"ENTER BIRTH DATE"	"ENT DT NAISSANCE"	"INGRESE FECHA NAC"
50	"ENTER DOB MMDDYY"	"NAISSANCE MMJJAA"	"INGRESE FDN MMDDAA"
51	"ENTER FLEET DATA"	"INFO PARC AUTO"	"INGRESE INFO DE FLOTA"
52	"ENTER REFERENCE"	"ENTREZ REFERENCE"	"INGRESE REFERENCIA"
53	"ENTER AUTH NUMBR"	"No AUTORISATION"	"INGRESE NUMERO AUT"
54	"ENTER HUB NUMBER"	"ENTREZ No NOYAU"	"INGRESE NRO HUB"
55	"ENTER HUBOMETER"	"COMPTEUR NOYAU"	"INGRESE MEDIDO DE HUB"
56	"ENTER TRAILER ID"	"ENT ID REMORQUE"	"INGRESE ID TRAILER"
57	"ODOMETER READING"	"LECTURE ODOMETRE"	"LECTURA ODOMETRO"
58	"REENTER ODOMETER"	"RE-ENT ODOMETRE"	"REINGRESE ODOMETRO"
59	"REENTER DRIV. ID"	"RE-ENT ID CONDUC"	"REINGRESE ID CHOFER"
60	"ENTER CUSTOMER ID"	"ENTREZ ID CLIENT"	"INGRESE ID CLIENTE"
61	"ENTER CUST. ID"	"ENTREZ ID CLIENT"	"INGRESE ID CLIENTE"
62	"ENTER ROUTE NUM"	"ENT No ROUTE"	"INGRESE NUM RUTA"
63	"ENTER FLEET NUM"	"ENT No PARC AUTO"	"INGRESE NUM FLOTA"
64	"FLEET PIN"	"NIP PARC AUTO"	"PIN DE FLOTA"
65	"DRIVER #"	"CONDUCTEUR"	"CONDUCTOR #"
66	"ENTER DRIVER #"	"ENT # CONDUCTEUR"	"INGRESE CONDUCTOR #"
67	"VEHICLE #"	"# VEHICULE"	"VEHICULO #"
68	"ENTER VEHICLE #"	"ENT # VEHICULE"	"INGRESE VEHICULO #"

69	"JOB #"	"# TRAVAIL"	"TRABAJO #"
70	"ENTER JOB #"	"ENTREZ # TRAVAIL"	"INGRESE TRABAJO #"
71	"DEPT NUMBER"	"No DEPARTEMENT"	"NUMERO DEPTO"
72	"DEPARTMENT #"	"DEPARTEMENT"	"DEPARTAMENTO #"
73	"ENTER DEPT #"	"ENT# DEPARTEMENT"	"INGRESE DEPTO #"
74	"LICENSE NUMBER"	"No PERMIS"	"NUMERO LICENCIA"
75	"LICENSE #"	"# PERMIS"	"LICENCIA #"
76	"ENTER LICENSE #"	"ENTREZ # PERMIS"	"INGRESE LICENCIA #"
77	"DATA"	"INFO"	"INFO"
78	"ENTER DATA"	"ENTREZ INFO"	"INGRESE INFO"
79	"CUSTOMER DATA"	"INFO CLIENT"	"INFO CLIENTE"
80	"ID #"	"# ID"	"ID #"
81	"ENTER ID #"	"ENTREZ # ID"	"INGRESE ID #"
82	"USER ID"	"ID UTILISATEUR"	"ID USUARIO"
83	"ROUTE #"	"# ROUTE"	"RUTA #"
84	"ENTER ROUTE #"	"ENTREZ # ROUTE"	"INGRESE RUTA #"
85	"ENTER CARD NUM"	"ENTREZ NO CARTE"	"INGRESE NUM TARJETA"
86	"EXP DATE(YYMM)"	"DATE EXPIR(AAMM)"	"FECHA EXP (AAMM)"
87	"PHONE NUMBER"	"NO TEL"	"NUMERO TELEFONO"
88	"CVV START DATE"	"CVV DATE DE DEBUT"	"CVV FECHA INICIO"
89	"ISSUE NUMBER"	"NO DEMISSION"	"NUMERO DE EMISION"
90	"START DATE (MMYY)"	"DATEDEDEBUT-AAMM"	"FECHA INICIO (AAMM)"

5.16 Multi-Language Support

In order to support multiple languages on the Vx 600BT Device, individual language files are required to be built and loaded onto the PIN pad at the time of configuration (see Appendix C PIN Pad Message Files for more information). The language or message files contain all of the pre-defined messages that can be displayed on the PIN pad screen when requested by the calling application (in this instance, the VeriFone Mobile Framework). Each message file has an identifying numeric code that represents the language translation used for that file. The currently defined country codes are listed below (additional codes need will require a new index value.

```
0  USA
1  FRANCE
2  GERMANY
3  UK
4  DENMARK
5  SWEDEN
6  ITALY
7  SPAIN
8  JAPAN
9  NORWAY
10  DENMARK 2
11  CHINA
```

It is up to the calling application to display prompts and other messages not defined in the PIN Pad Message File in the desired language. The PIN Pad Message File messages are used only for those command requests that specify the Message ID of the desired message. It is also the responsibility of the application to handle any language specific text display and prompting on the device in the appropriate language.

5.17 Country Specific Implementation Considerations

Implementation of the Vx 600BT Device utilizing XPI and the VeriFone Mobile Framework will require coordination to ensure that all country-specific implementation requirements are met with regards to both hardware and software

certifications. Several factors that should be considered are:

- Identify any hardware certifications required for the Vx 600BT device
- Determine in-country payment types required, such as generic payments, EMV specifics, local card types; and identify what is in place for these.
- Identify solutions certification needs, specifically with the Middleware and Processor; a definition of the processes that need to be followed and requirements that must be met for final certification
- Any PIN pad software certification needs that are specific to PIN pad usage
- EMV-Kernel requirements, configurations, and certification needs; including any country specific EMV libraries and/or software implementations
- Key loading, key injection requirements for the device (local payment types and EMV keys).
- Determine in-country payment flows; particularly, any specifics for local card processing needs.
- Determine in-country payment expertise to assist in guiding through the process

5.18 Troubleshooting

Debugger terminates unexpectedly during a device update

If you receive a message similar to the below during a XCode debugger session while working with the e210/e255/e315 PINPad and executing a download or parameter update to the device, please note that this error occurs because the e210/e255/e315 received an EOT after downloading all of the files, which reboots it. Similarly, during XCode debug session, if you unplug the device, it will abruptly exit the debug session. That is what is occurring in this instance which is the same as if you unplugged the e210/e255/e315 in the middle of a debug session. The software is trying to do clean-up, but the device is being rebooted, so you get the stream write error, followed by the SIGTERM as a result of the disconnecting stream.

```
2010-12-16 10:21:59.271 e210/e255/e315 Update Utility[85:307] ERROR - /SourceCache/ExternalAccessory/
      ExternalAccessory- 102/EAOutputStream.m:-[EAOutputStream write:maxLength:] - 233 failed to write because stream
      does not have space available
Watchdog has expired. Remote device was disconnected? Debugging session terminated. Watchdog has expired.
      Remote device was disconnected? Debugging session terminated.
The Debugger has exited due to signal 15 (SIGTERM).The Debugger has exited due to signal 15 (SIGTERM).
```

Application not installed

Would you like to install message when running the application. If you receive this message, your application has not registered support for the e210/e255/e315 PINPad through the EAAccessoryProtocol. Refer to - Add Protocol Strings to .pList

Chapter 6

Deprecated List

Member [VFIBarcode setBeep:]

Please see configureBeep:()

Member [VFIPinpad dataEntryRequest:language:tip:cashback:surcharge:]

Use combinedPromptCommand:language:transID:(menuArray:surcharge:accountNum:ipOptions:cashBackOptions:surchargeOptions:()

Member [VFIPinpad dataEntryRequestPrompt:]

Use combinedPromptCommand:language:transID:(menuArray:surcharge:accountNum:ipOptions:cashBackOptions:surchargeOptions:()

Member [VFIPinpad displayMessage:clearScreen:]

Use displayMessages:message1:message2:clearScreen:()

Member [VFIPinpad enableMSR]

Replaced by enableMSRDualTrack()

Member [VFIPinpad Q40]

Replaced by Q42()

Member [VFIPinpad Z2:clearScreen:]

Use displayMessages:message1:message2:clearScreen:()

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<AsyncSocketDelegate> . . . 142
CMD_DEF . . . 145
<EAAccessoryDelegate>
  VFIBarcode . . . 157
  VFIControl . . . 195
  VFIPinpad . . . 246
HOST_RX_BUF . . . 147
HOST_TX_BUF . . . 147
M_BEEP_DEF . . . 149
NSObject
  AsyncSocket . . . 131
  decryptedCredentials . . . 145
  decryptedCredentials . . . 145
  ISCP_HIGH . . . 148
  swipeData . . . 152
  swipeData . . . 152
  VFIAccessoryMgr . . . 155
  VFIBarcode . . . 157
  VFIBTBridge . . . 188
  VFICardData . . . 190
  VFICipheredData . . . 193
  VFIControl . . . 195
  VFIDataEntry . . . 216
  VFIDiagnostics . . . 219
  VFIEMAFiles . . . 221
  VFIEMVAuthorization . . . 223
  VFIEMVCompletionData . . . 224
  VFIEMVConfiguration . . . 227
  VFIEMVResponse . . . 233
  VFIEMVTags . . . 236
  VFIEncryptedBarcode . . . 237
  VFIEncryptionData . . . 239
  VFIFinancialData . . . 240
  VFIGen3DeviceInfo . . . 242

VFIKeypadVersion 245
VFIPinpad 246
VFISoftwareVersion 368
VFIStream 370
<NSObject>
<VFIBarcodeDelegate> 183
<VFIControlDelegate> 212
<VFIPinpadDelegate> 362
<VFIStreamDelegate> 372
VFIAccessoryMgr 155
<NSStreamDelegate>
VFIBarcode 157
VFIControl 195
VFIPinpad 246
VFIStream 370
PARM_DEF 150
POWER_STATUS 151
symb_INFO 153
T_BEEP 154
T_BEEP_DEF 154
<UIApplicationDelegate>
VFIBarcode 157
VFIControl 195
VFIPinpad 246
<VFIZontalkDelegate>
VFIPinpad 246

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AsyncSocket 131
<AsyncSocketDelegate> 142
CMD_DEF 145
decryptedCredentials
Encapsulating data class utilized by VFIConnect 145
HOST_RX_BUF 147
HOST_TX_BUF 147
ISCP_HIGH
Encapsulating data class utilized by VFIBarcode 148
M_BEEP_DEF 149
PARM_DEF 150
POWER_STATUS
POWER_STATUS structure 151
swipeData
Encapsulating data class utilized by VFIConnect 152
symb_INFO 153
T_BEEP 154
T_BEEP_DEF 154
VFIAccessoryMgr 155
VFIBarcode
API methods for e210/e255/e315 Pinpad Barcode Scanner 157
<VFIBarcodeDelegate>
Protocol methods established for VFIBarcode class 183
VFIBTBridge 188
VFICardData
Encapsulating data class utilized by VFIPinpad 190
VFICipheredData
Encapsulating data class utilized by VFIPinpad 193
VFIControl
API methods for e210/e255/e315 Pinpad Control 195
<VFIControlDelegate>
Protocol methods established for VFIControl class 212
VFIDataEntry
Encapsulating data class utilized by VFIPinpad 216

VFIDiagnostics
Encapsulating data class utilized by VFIPinpad 219
VFIEMAFiles
Encapsulating data class utilized by VFIPinpad 221
VFIEMVAuthorization
Encapsulating data class utilized by VFIPinpad 223
VFIEMVCompletionData
Encapsulating data class utilized by VFIPinpad 224
VFIEMVConfiguration
Encapsulating data class utilized by VFIPinpad 227
VFIEMVResponse
Encapsulating data class utilized by VFIPinpad 233
VFIEMVTags
Encapsulating data class utilized by VFIPinpad 236
VFIEncryptedBarcode
Encapsulating data class representing Encrypted Barcode Data 237
VFIEncryptionData
Encapsulating data class utilized by VFIPinpad 239
VFIFinancialData
Encapsulating data class utilized by VFIPinpad 240
VFIGen3DeviceInfo
Encapsulating data class utilized by VFIPinpad 242
VFIKeypadVersion
Encapsulating data class utilized by VFIControl 245
VFIPinpad
API methods for e210/e255/e315 for XPI 246
<VFIPinpadDelegate>
Protocol methods established for VFIPinpad class NEW 362
VFISoftwareVersion
Encapsulating data class utilized by VFIControl 368
VFIStream 370
<VFIStreamDelegate> 372

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

AsyncSocket.h . . . 375
Barcode_host.h . . . 376
decryptedCredentials.h . . . 395
GlobalMarkup.d . . . 395
ISCP_HIGH.h . . . 396
swipeData.h . . . 396
VFIAccessoryMgr.h . . . 397
VFIBarcode.h . . . 402
VFIBTBridge.h . . . 405
VFICardData.h . . . 405
VFICipheredData.h . . . 405
VFIControl.h . . . 406
VFIDataEntry.h . . . 406
VFIDecryptedCredentials.h . . . 406
VFIDiagnostics.h . . . 406
VFIEMAFiles.h . . . 407
VFIEMVAuthorization.h . . . 407
VFIEMVCompletionData.h . . . 407
VFIEMVConfiguration.h . . . 407
VFIEMVResponse.h . . . 408
VFIEMVTags.h . . . 408
VFIEncryptedBarcode.h . . . 408
VFIEncryptionData.h . . . 408
VFIFinancialData.h . . . 409
VFIGen3DeviceInfo.h . . . 409
VFIKeypadVersion.h . . . 409
VFIPinpad.h . . . 409
VFISoftwareVersion.h . . . 411
VFIStream.h . . . 412
VFISwipeData.h . . . 412

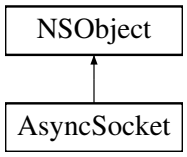
Chapter 10

Class Documentation

10.1 AsyncSocket Class Reference

`#import <AsyncSocket.h>`

Inheritance diagram for AsyncSocket:



Instance Methods

- (id) - init
- (id) - initWithDelegate:
- (id) - initWithDelegate:userData:
- (NSString *) - description
- (void) - write:
- (id) - delegate

  Use "canSafelySetDelegate" to see if there is any pending business (reads and writes) with the current delegate before changing it.
- (BOOL) - canSafelySetDelegate
- (void) - setDelegate:
- (long) - userData
- (void) - setUserData:
- (CFSocketRef) - getCFSocket
- (CFReadStreamRef) - getCFReadStream
- (CFWriteStreamRef) - getCFWriteStream
- (BOOL) - acceptOnPort:error:

  Tells the socket to begin listening and accepting connections on the given port.
- (BOOL) - acceptOnInterface:port:error:

  This method is the same as acceptOnPort:error: with the additional option of specifying which interface to listen on.
- (BOOL) - connectToHost:onPort:error:

Connects to the given host and port.

- (BOOL) - connectToHost:onPort:withTimeout:error:

  This method is the same as connectToHost:onPort:error: with an additional timeout option.
- (BOOL) - connectToAddress:error:

  Connects to the given address, specified as a sockaddr structure wrapped in a NSData object.
- (BOOL) - connectToAddress:withTimeout:error:

  This method is the same as connectToAddress:error: with an additional timeout option.
- (BOOL) - connectToAddress:viaInterfaceAddress:withTimeout:error:
- (void) - disconnect

  Disconnects immediately.
- (void) - disconnectAfterReading

  Disconnects after all pending reads have completed.
- (void) - disconnectAfterWriting

  Disconnects after all pending writes have completed.
- (void) - disconnectAfterReadingAndWriting

  Disconnects after all pending reads and writes have completed.
- (BOOL) - isConnected
- (NSString ∗) - connectedHost

  Returns the local or remote host and port to which this socket is connected, or nil and 0 if not connected.
- (UInt16) - connectedPort
- (NSString ∗) - localHost
- (UInt16) - localPort
- (NSData ∗) - connectedAddress

  Returns the local or remote address to which this socket is connected, specified as a sockaddr structure wrapped in a NSData object.
- (NSData ∗) - localAddress
- (BOOL) - isIPv4

  Returns whether the socket is IPv4 or IPv6.
- (BOOL) - isIPv6
- (void) - readDataWithTimeout:tag:

  Reads the first available bytes that become available on the socket.
- (void) - readDataWithTimeout:buffer:bufferOffset:tag:

  Reads the first available bytes that become available on the socket.
- (void) - readDataWithTimeout:buffer:bufferOffset:maxLength:tag:

  Reads the first available bytes that become available on the socket.
- (void) - readDataToLength:withTimeout:tag:

  Reads the given number of bytes.
- (void) - readDataToLength:withTimeout:buffer:bufferOffset:tag:

  Reads the given number of bytes.
- (void) - readDataToData:withTimeout:tag:

  Reads bytes until (and including) the passed "data" parameter, which acts as a separator.
- (void) - readDataToData:withTimeout:buffer:bufferOffset:tag:

  Reads bytes until (and including) the passed "data" parameter, which acts as a separator.
- (void) - readDataToData:withTimeout:maxLength:tag:

  Reads bytes until (and including) the passed "data" parameter, which acts as a separator.
- (void) - readDataToData:withTimeout:buffer:bufferOffset:maxLength:tag:

  Reads bytes until (and including) the passed "data" parameter, which acts as a separator.
- (void) - writeData:withTimeout:tag:

Writes data to the socket, and calls the delegate when finished.

- (float) - progressOfReadReturningTag:bytesDone:total:

  Returns progress of current read or write, from 0.0 to 1.0, or NaN if no read/write (use isnan() to check).
- (float) - progressOfWriteReturningTag:bytesDone:total:
- (void) - startTLS:

  Secures the connection using SSL/TLS.
- (void) - enablePreBuffering

  For handling readDataToData requests, data is necessarily read from the socket in small increments.
- (BOOL) - moveToRunLoop:

  When you create an AsyncSocket, it is added to the runloop of the current thread.
- (BOOL) - setRunLoopModes:

  Allows you to configure which run loop modes the socket uses.
- (BOOL) - addRunLoopMode:
- (BOOL) - removeRunLoopMode:
- (NSArray ∗) - runLoopModes

  Returns the current run loop modes the AsyncSocket instance is operating in.
- (NSData ∗) - unreadData

  In the event of an error, this method may be called during onSocket:willDisconnectWithError: to read any data that's left on the socket.

Class Methods

- (NSData ∗) + CRLFData
- (NSData ∗) + CRData
- (NSData ∗) + LFData
- (NSData ∗) + ZeroData

Protected Attributes

- CFSocketNativeHandle theNativeSocket4
- CFSocketNativeHandle theNativeSocket6
- CFSocketRef theSocket4
- CFSocketRef theSocket6
- CFReadStreamRef theReadStream
- CFWriteStreamRef theWriteStream
- CFRunLoopSourceRef theSource4
- CFRunLoopSourceRef theSource6
- CFRunLoopRef theRunLoop
- CFSocketContext theContext
- NSArray ∗ theRunLoopModes
- NSTimer ∗ theConnectTimer
- NSMutableArray ∗ theReadQueue
- AsyncReadPacket ∗ theCurrentRead
- NSTimer ∗ theReadTimer
- NSMutableData ∗ partialReadBuffer
- NSMutableArray ∗ theWriteQueue
- AsyncWritePacket ∗ theCurrentWrite
- NSTimer ∗ theWriteTimer
- id theDelegate
- UInt16 theFlags
- long theUserData

10.1.1 Method Documentation

10.1.1.1 - (BOOL) acceptOnInterface: (NSString ∗) *interface* port:(UInt16) *port* error:(NSError ∗∗) *errPtr*

This method is the same as acceptOnPort:error: with the additional option of specifying which interface to listen on.

So, for example, if you were writing code for a server that has multiple IP addresses, you could specify which address you wanted to listen on. Or you could use it to specify that the socket should only accept connections over ethernet, and not other interfaces such as wifi. You may also use the special strings "localhost" or "loopback" to specify that the socket only accept connections from the local machine.

To accept connections on any interface pass nil, or simply use the acceptOnPort:error: method.

10.1.1.2 - (BOOL) acceptOnPort: (UInt16) *port* error:(NSError ∗∗) *errPtr*

Tells the socket to begin listening and accepting connections on the given port.

When a connection comes in, the AsyncSocket instance will call the various delegate methods (see above). The socket will listen on all available interfaces (e.g. wifi, ethernet, etc)

10.1.1.3 - (BOOL) addRunLoopMode: (NSString ∗) *runLoopMode*

10.1.1.4 - (BOOL) canSafelySetDelegate

10.1.1.5 - (NSData ∗) connectedAddress

Returns the local or remote address to which this socket is connected, specified as a sockaddr structure wrapped in a NSData object.

See also the connectedHost, connectedPort, localHost and localPort methods.

10.1.1.6 - (NSString ∗) connectedHost

Returns the local or remote host and port to which this socket is connected, or nil and 0 if not connected.

The host will be an IP address.

10.1.1.7 - (UInt16) connectedPort

10.1.1.8 - (BOOL) connectToAddress: (NSData ∗) *remoteAddr* error:(NSError ∗∗) *errPtr*

Connects to the given address, specified as a sockaddr structure wrapped in a NSData object.

For example, a NSData object returned from NSNetService's addresses method.

If you have an existing struct sockaddr you can convert it to a NSData object like so: struct sockaddr sa -> NSData ∗dsa = [NSData dataWithBytes:&remoteAddr length:remoteAddr.sa_len]; struct sockaddr ∗sa -> NSData ∗dsa = [NSData dataWithBytes:remoteAddr length:remoteAddr->sa_len];

10.1.1.9 - (BOOL) connectToAddress: (NSData ∗) *remoteAddr* viaInterfaceAddress:(NSData ∗) *interfaceAddr* withTimeout:(NSTimeInterval) *timeout* error:(NSError ∗∗) *errPtr*

10.1.1.10 - (BOOL) connectToAddress: (NSData ∗) *remoteAddr* withTimeout:(NSTimeInterval) *timeout* error:(NSError ∗∗) *errPtr*

This method is the same as connectToAddress:error: with an additional timeout option.

To not time out use a negative time interval, or simply use the connectToAddress:error: method.

10.1.1.11 - (BOOL) connectToHost: (NSString ∗) *hostname* onPort:(UInt16) *port* error:(NSError ∗∗) *errPtr*

Connects to the given host and port.

The host may be a domain name (e.g. "deusty.com") or an IP address string (e.g. "192.168.0.2")

10.1.1.12 - (BOOL) connectToHost: (NSString ∗) *hostname* onPort:(UInt16) *port* withTimeout:(NSTimeInterval) *timeout* error:(NSError ∗∗) *errPtr*

This method is the same as connectToHost:onPort:error: with an additional timeout option.

To not time out use a negative time interval, or simply use the connectToHost:onPort:error: method.

10.1.1.13 + (NSData ∗) CRData

10.1.1.14 + (NSData ∗) CRLFData

10.1.1.15 - (id) delegate

Use "canSafelySetDelegate" to see if there is any pending business (reads and writes) with the current delegate before changing it.

It is, of course, safe to change the delegate before connecting or accepting connections.

10.1.1.16 - (NSString ∗) description

10.1.1.17 - (void) disconnect

Disconnects immediately.

Any pending reads or writes are dropped. If the socket is not already disconnected, the onSocketDidDisconnect delegate method will be called immediately, before this method returns.

Please note the recommended way of releasing an AsyncSocket instance (e.g. in a dealloc method) [asyncSocket setDelegate:nil]; [asyncSocket disconnect]; [asyncSocket release];

10.1.1.18 - (void) disconnectAfterReading

Disconnects after all pending reads have completed.

After calling this, the read and write methods will do nothing. The socket will disconnect even if there are still pending writes.

10.1.1.19 - (void) disconnectAfterReadingAndWriting

Disconnects after all pending reads and writes have completed.

After calling this, the read and write methods will do nothing.

10.1.1.20 - (void) disconnectAfterWriting

Disconnects after all pending writes have completed.

After calling this, the read and write methods will do nothing. The socket will disconnect even if there are still pending reads.

10.1.1.21 - (void) enablePreBuffering

For handling readDataToData requests, data is necessarily read from the socket in small increments.

The performance can be much improved by allowing AsyncSocket to read larger chunks at a time and store any overflow in a small internal buffer. This is termed pre-buffering, as some data may be read for you before you ask for it. If you use readDataToData a lot, enabling pre-buffering will result in better performance, especially on the iPhone.

The default pre-buffering state is controlled by the DEFAULT_PREBUFFERING definition. It is highly recommended one leave this set to YES.

This method exists in case pre-buffering needs to be disabled by default for some unforeseen reason. In that case, this method exists to allow one to easily enable pre-buffering when ready.

10.1.1.22 - (CFReadStreamRef) getCFReadStream

10.1.1.23 - (CFSocketRef) getCFSocket

10.1.1.24 - (CFWriteStreamRef) getCFWriteStream

10.1.1.25 - (id) init

10.1.1.26 - (id) initWithDelegate: (id) *delegate*

10.1.1.27 - (id) initWithDelegate: (id) *delegate* userData:(long) *userData*

10.1.1.28 - (BOOL) isConnected

10.1.1.29 - (BOOL) isIPv4

Returns whether the socket is IPv4 or IPv6.

An accepting socket may be both.

10.1.1.30 - (BOOL) isIPv6

10.1.1.31 + (NSData ∗) LFData

10.1.1.32 - (NSData ∗) localAddress

10.1.1.33 - (NSString ∗) localHost

10.1.1.34 - (UInt16) localPort

10.1.1.35 - (BOOL) moveToRunLoop: (NSRunLoop ∗) *runLoop*

When you create an AsyncSocket, it is added to the runloop of the current thread.

So for manually created sockets, it is easiest to simply create the socket on the thread you intend to use it.

If a new socket is accepted, the delegate method onSocket:wantsRunLoopForNewSocket: is called to allow you to place the socket on a separate thread. This works best in conjunction with a thread pool design.

If, however, you need to move the socket to a separate thread at a later time, this method may be used to accomplish the task.

This method must be called from the thread/runloop the socket is currently running on.

Note: After calling this method, all further method calls to this object should be done from the given runloop. Also, all delegate calls will be sent on the given runloop.

10.1.1.36 - (float) progressOfReadReturningTag: (long *) *tag* bytesDone:(NSUInteger *) *done* total:(NSUInteger *) *total*

Returns progress of current read or write, from 0.0 to 1.0, or NaN if no read/write (use isnan() to check).

"tag", "done" and "total" will be filled in if they aren't NULL.

10.1.1.37 - (float) progressOfWriteReturningTag: (long *) *tag* bytesDone:(NSUInteger *) *done* total:(NSUInteger *) *total*

10.1.1.38 - (void) readDataToData: (NSData *) *data* withTimeout:(NSTimeInterval) *timeout* buffer:(NSMutableData *) *buffer* bufferOffset:(NSUInteger) *offset* maxLength:(NSUInteger) *length* tag:(long) *tag*

Reads bytes until (and including) the passed "data" parameter, which acts as a separator.

The bytes will be appended to the given byte buffer starting at the given offset. The given buffer will automatically be increased in size if needed. A maximum of length bytes will be read.

If the timeout value is negative, the read operation will not use a timeout. If the buffer if nil, a buffer will automatically be created for you.

If maxLength is zero, no length restriction is enforced. Otherwise if maxLength bytes are read without completing the read, it is treated similarly to a timeout - the socket is closed with a AsyncSocketReadMaxedOutError. The read will complete successfully if exactly maxLength bytes are read and the given data is found at the end.

If you pass a maxLength parameter that is less than the length of the data parameter, the method will do nothing, and the delegate will not be called. If the bufferOffset is greater than the length of the given buffer, the method will do nothing, and the delegate will not be called.

If you pass a buffer, you must not alter it in any way while AsyncSocket is using it. After completion, the data returned in onSocket:didReadData:withTag: will be a subset of the given buffer. That is, it will reference the bytes that were appended to the given buffer.

To read a line from the socket, use the line separator (e.g. CRLF for HTTP, see below) as the "data" parameter. Note that this method is not character-set aware, so if a separator can occur naturally as part of the encoding for a character, the read will prematurely end.

10.1.1.39 - (void) readDataToData: (NSData *) *data* withTimeout:(NSTimeInterval) *timeout* buffer:(NSMutableData *) *buffer* bufferOffset:(NSUInteger) *offset* tag:(long) *tag*

Reads bytes until (and including) the passed "data" parameter, which acts as a separator.

The bytes will be appended to the given byte buffer starting at the given offset. The given buffer will automatically be increased in size if needed.

If the timeout value is negative, the read operation will not use a timeout. If the buffer if nil, a buffer will automatically be created for you.

If the bufferOffset is greater than the length of the given buffer, the method will do nothing, and the delegate will not be called.

If you pass a buffer, you must not alter it in any way while AsyncSocket is using it. After completion, the data returned in onSocket:didReadData:withTag: will be a subset of the given buffer. That is, it will reference the bytes that were appended to the given buffer.

To read a line from the socket, use the line separator (e.g. CRLF for HTTP, see below) as the "data" parameter. Note that this method is not character-set aware, so if a separator can occur naturally as part of the encoding for a character, the read will prematurely end.

10.1.1.40 - (void) readDataToData: (NSData *) *data* withTimeout:(NSTimeInterval) *timeout* maxLength:(NSUInteger) *length* tag:(long) *tag*

Reads bytes until (and including) the passed "data" parameter, which acts as a separator.

If the timeout value is negative, the read operation will not use a timeout.

If maxLength is zero, no length restriction is enforced. Otherwise if maxLength bytes are read without completing the read, it is treated similarly to a timeout - the socket is closed with a AsyncSocketReadMaxedOutError. The read will complete successfully if exactly maxLength bytes are read and the given data is found at the end.

If you pass nil or zero-length data as the "data" parameter, the method will do nothing, and the delegate will not be called. If you pass a maxLength parameter that is less than the length of the data parameter, the method will do nothing, and the delegate will not be called.

To read a line from the socket, use the line separator (e.g. CRLF for HTTP, see below) as the "data" parameter. Note that this method is not character-set aware, so if a separator can occur naturally as part of the encoding for a character, the read will prematurely end.

10.1.1.41 - (void) readDataToData: (NSData *) *data* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Reads bytes until (and including) the passed "data" parameter, which acts as a separator.

If the timeout value is negative, the read operation will not use a timeout.

If you pass nil or zero-length data as the "data" parameter, the method will do nothing, and the delegate will not be called.

To read a line from the socket, use the line separator (e.g. CRLF for HTTP, see below) as the "data" parameter. Note that this method is not character-set aware, so if a separator can occur naturally as part of the encoding for a character, the read will prematurely end.

10.1.1.42 - (void) readDataToLength: (NSUInteger) *length* withTimeout:(NSTimeInterval) *timeout* buffer:(NSMutableData *) *buffer* bufferOffset:(NSUInteger) *offset* tag:(long) *tag*

Reads the given number of bytes.

The bytes will be appended to the given byte buffer starting at the given offset. The given buffer will automatically be increased in size if needed.

If the timeout value is negative, the read operation will not use a timeout. If the buffer if nil, a buffer will automatically be created for you.

If the length is 0, this method does nothing and the delegate is not called. If the bufferOffset is greater than the length of the given buffer, the method will do nothing, and the delegate will not be called.

If you pass a buffer, you must not alter it in any way while AsyncSocket is using it. After completion, the data returned in onSocket:didReadData:withTag: will be a subset of the given buffer. That is, it will reference the bytes that were

appended to the given buffer.

10.1.1.43 - (void) readDataToLength: (NSUInteger) *length* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Reads the given number of bytes.

If the timeout value is negative, the read operation will not use a timeout.

If the length is 0, this method does nothing and the delegate is not called.

10.1.1.44 - (void) readDataWithTimeout: (NSTimeInterval) *timeout* buffer:(NSMutableData ∗) *buffer* bufferOffset:(NSUInteger) *offset* maxLength:(NSUInteger) *length* tag:(long) *tag*

Reads the first available bytes that become available on the socket.

The bytes will be appended to the given byte buffer starting at the given offset. The given buffer will automatically be increased in size if needed. A maximum of length bytes will be read.

If the timeout value is negative, the read operation will not use a timeout. If the buffer if nil, a buffer will automatically be created for you. If maxLength is zero, no length restriction is enforced.

If the bufferOffset is greater than the length of the given buffer, the method will do nothing, and the delegate will not be called.

If you pass a buffer, you must not alter it in any way while AsyncSocket is using it. After completion, the data returned in onSocket:didReadData:withTag: will be a subset of the given buffer. That is, it will reference the bytes that were appended to the given buffer.

10.1.1.45 - (void) readDataWithTimeout: (NSTimeInterval) *timeout* buffer:(NSMutableData ∗) *buffer* bufferOffset:(NSUInteger) *offset* tag:(long) *tag*

Reads the first available bytes that become available on the socket.

The bytes will be appended to the given byte buffer starting at the given offset. The given buffer will automatically be increased in size if needed.

If the timeout value is negative, the read operation will not use a timeout. If the buffer if nil, the socket will create a buffer for you.

If the bufferOffset is greater than the length of the given buffer, the method will do nothing, and the delegate will not be called.

If you pass a buffer, you must not alter it in any way while AsyncSocket is using it. After completion, the data returned in onSocket:didReadData:withTag: will be a subset of the given buffer. That is, it will reference the bytes that were appended to the given buffer.

10.1.1.46 - (void) readDataWithTimeout: (NSTimeInterval) *timeout* tag:(long) *tag*

Reads the first available bytes that become available on the socket.

If the timeout value is negative, the read operation will not use a timeout.

10.1.1.47 - (BOOL) removeRunLoopMode: (NSString ∗) *runLoopMode*

10.1.1.48 - (NSArray ∗) runLoopModes

Returns the current run loop modes the AsyncSocket instance is operating in.

The default set of run loop modes is NSDefaultRunLoopMode.

10.1.1.49 - (void) setDelegate: (id) *delegate*

10.1.1.50 - (BOOL) setRunLoopModes: (NSArray ∗) *runLoopModes*

Allows you to configure which run loop modes the socket uses.

The default set of run loop modes is NSDefaultRunLoopMode.

If you'd like your socket to continue operation during other modes, you may want to add modes such as NSModalPanelRunLoopMode or NSEventTrackingRunLoopMode. Or you may simply want to use NSRunLoopCommonModes.

Accepted sockets will automatically inherit the same run loop modes as the listening socket.

Note: NSRunLoopCommonModes is defined in 10.5. For previous versions one can use kCFRunLoopCommonModes.

10.1.1.51 - (void) setUserData: (long) *userData*

10.1.1.52 - (void) startTLS: (NSDictionary ∗) *tlsSettings*

Secures the connection using SSL/TLS.

This method may be called at any time, and the TLS handshake will occur after all pending reads and writes are finished. This allows one the option of sending a protocol dependent StartTLS message, and queuing the upgrade to TLS at the same time, without having to wait for the write to finish. Any reads or writes scheduled after this method is called will occur over the secured connection.

The possible keys and values for the TLS settings are well documented. Some possible keys are:

- kCFStreamSSLLevel
- kCFStreamSSLAllowsExpiredCertificates
- kCFStreamSSLAllowsExpiredRoots
- kCFStreamSSLAllowsAnyRoot
- kCFStreamSSLValidatesCertificateChain
- kCFStreamSSLPeerName
- kCFStreamSSLCertificates
- kCFStreamSSLIsServer

Please refer to Apple's documentation for associated values, as well as other possible keys.

If you pass in nil or an empty dictionary, the default settings will be used.

The default settings will check to make sure the remote party's certificate is signed by a trusted 3rd party certificate agency (e.g. verisign) and that the certificate is not expired. However it will not verify the name on the certificate unless you give it a name to verify against via the kCFStreamSSLPeerName key. The security implications of this are important to understand. Imagine you are attempting to create a secure connection to MySecureServer.com, but your socket

gets directed to MaliciousServer.com because of a hacked DNS server. If you simply use the default settings, and MaliciousServer.com has a valid certificate, the default settings will not detect any problems since the certificate is valid. To properly secure your connection in this particular scenario you should set the kCFStreamSSLPeerName property to "MySecureServer.com". If you do not know the peer name of the remote host in advance (for example, you're not sure if it will be "domain.com" or "www.domain.com"), then you can use the default settings to validate the certificate, and then use the X509Certificate class to verify the issuer after the socket has been secured. The X509Certificate class is part of the CocoaAsyncSocket open source project.

10.1.1.53 - (NSData ∗) unreadData

In the event of an error, this method may be called during onSocket:willDisconnectWithError: to read any data that's left on the socket.

10.1.1.54 - (long) userData

10.1.1.55 - (void) write: (NSData ∗) *data*

10.1.1.56 - (void) writeData: (NSData ∗) *data* withTimeout:(NSTimeInterval) *timeout* tag:(long) *tag*

Writes data to the socket, and calls the delegate when finished.

If you pass in nil or zero-length data, this method does nothing and the delegate will not be called. If the timeout value is negative, the write operation will not use a timeout.

10.1.1.57 + (NSData ∗) ZeroData

10.1.2 Member Data Documentation

10.1.2.1 - (NSMutableData∗) partialReadBuffer [protected]

10.1.2.2 - (NSTimer∗) theConnectTimer [protected]

10.1.2.3 - (CFSocketContext) theContext [protected]

10.1.2.4 - (AsyncReadPacket∗) theCurrentRead [protected]

10.1.2.5 - (AsyncWritePacket∗) theCurrentWrite [protected]

10.1.2.6 - (id) theDelegate [protected]

10.1.2.7 - (UInt16) theFlags [protected]

10.1.2.8 - (CFSocketNativeHandle) theNativeSocket4 [protected]

10.1.2.9 - (CFSocketNativeHandle) theNativeSocket6 [protected]

10.1.2.10 - (NSMutableArray∗) theReadQueue [protected]

10.1.2.11 - (CFReadStreamRef) theReadStream [protected]

10.1.2.12 - (NSTimer∗) theReadTimer [protected]

10.1.2.13 - (CFRunLoopRef) theRunLoop [protected]

10.1.2.14 - (NSArray*) theRunLoopModes [protected]

10.1.2.15 - (CFSocketRef) theSocket4 [protected]

10.1.2.16 - (CFSocketRef) theSocket6 [protected]

10.1.2.17 - (CFRunLoopSourceRef) theSource4 [protected]

10.1.2.18 - (CFRunLoopSourceRef) theSource6 [protected]

10.1.2.19 - (long) theUserData [protected]

10.1.2.20 - (NSMutableArray*) theWriteQueue [protected]

10.1.2.21 - (CFWriteStreamRef) theWriteStream [protected]

10.1.2.22 - (NSTimer*) theWriteTimer [protected]

The documentation for this class was generated from the following file:

- AsyncSocket.h

10.2 <AsyncSocketDelegate> Protocol Reference

```
#import <AsyncSocket.h>
```

Instance Methods

- (void) - onSocket:willDisconnectWithError:

  In the event of an error, the socket is closed.
- (void) - onSocketDidDisconnect:

  Called when a socket disconnects with or without error.
- (void) - onSocket:didAcceptNewSocket:

  Called when a socket accepts a connection.
- (NSRunLoop *) - onSocket:wantsRunLoopForNewSocket:

  Called when a new socket is spawned to handle a connection.
- (BOOL) - onSocketWillConnect:

  Called when a socket is about to connect.
- (void) - onSocket:didConnectToHost:port:

  Called when a socket connects and is ready for reading and writing.
- (void) - onSocket:didReadData:withTag:

  Called when a socket has completed reading the requested data into memory.
- (void) - onSocket:didReadPartialDataOfLength:tag:

  Called when a socket has read in data, but has not yet completed the read.
- (void) - onSocket:didWriteDataWithTag:

  Called when a socket has completed writing the requested data.

- (void) - onSocket:didWritePartialDataOfLength:tag:

  Called when a socket has written some data, but has not yet completed the entire write.
- (NSTimeInterval) - onSocket:shouldTimeoutReadWithTag:elapsed:bytesDone:

  Called if a read operation has reached its timeout without completing.
- (NSTimeInterval) - onSocket:shouldTimeoutWriteWithTag:elapsed:bytesDone:

  Called if a write operation has reached its timeout without completing.
- (void) - onSocketDidSecure:

  Called after the socket has successfully completed SSL/TLS negotiation.

10.2.1 Method Documentation

10.2.1.1 - (void) onSocket: (AsyncSocket *) *sock* didAcceptNewSocket:(AsyncSocket *) *newSocket* [optional]

Called when a socket accepts a connection.

Another socket is spawned to handle it. The new socket will have the same delegate and will call "onSocket:didConnectToHost:port:".

10.2.1.2 - (void) onSocket: (AsyncSocket *) *sock* didConnectToHost:(NSString *) *host* port:(UInt16) *port* [optional]

Called when a socket connects and is ready for reading and writing.

The host parameter will be an IP address, not a DNS name.

10.2.1.3 - (void) onSocket: (AsyncSocket *) *sock* didReadData:(NSData *) *data* withTag:(long) *tag* [optional]

Called when a socket has completed reading the requested data into memory.

Not called if there is an error.

10.2.1.4 - (void) onSocket: (AsyncSocket *) *sock* didReadPartialDataOfLength:(NSUInteger) *partialLength* tag:(long) *tag* [optional]

Called when a socket has read in data, but has not yet completed the read.

This would occur if using readToData: or readToLength: methods. It may be used to for things such as updating progress bars.

10.2.1.5 - (void) onSocket: (AsyncSocket *) *sock* didWriteDataWithTag:(long) *tag* [optional]

Called when a socket has completed writing the requested data.

Not called if there is an error.

10.2.1.6 - (void) onSocket: (AsyncSocket *) *sock* didWritePartialDataOfLength:(NSUInteger) *partialLength* tag:(long) *tag* [optional]

Called when a socket has written some data, but has not yet completed the entire write.

It may be used to for things such as updating progress bars.

10.2.1.7 - (NSTimeInterval) onSocket: (AsyncSocket *) *sock* shouldTimeoutReadWithTag:(long) *tag* elapsed:(NSTimeInterval) *elapsed* bytesDone:(NSUInteger) *length* [optional]

Called if a read operation has reached its timeout without completing.

This method allows you to optionally extend the timeout. If you return a positive time interval (> 0) the read's timeout will be extended by the given amount. If you don't implement this method, or return a non-positive time interval ($<= 0$) the read will timeout as usual.

The elapsed parameter is the sum of the original timeout, plus any additions previously added via this method. The length parameter is the number of bytes that have been read so far for the read operation.

Note that this method may be called multiple times for a single read if you return positive numbers.

10.2.1.8 - (NSTimeInterval) onSocket: (AsyncSocket *) *sock* shouldTimeoutWriteWithTag:(long) *tag* elapsed:(NSTimeInterval) *elapsed* bytesDone:(NSUInteger) *length* [optional]

Called if a write operation has reached its timeout without completing.

This method allows you to optionally extend the timeout. If you return a positive time interval (> 0) the write's timeout will be extended by the given amount. If you don't implement this method, or return a non-positive time interval ($<= 0$) the write will timeout as usual.

The elapsed parameter is the sum of the original timeout, plus any additions previously added via this method. The length parameter is the number of bytes that have been written so far for the write operation.

Note that this method may be called multiple times for a single write if you return positive numbers.

10.2.1.9 - (NSRunLoop *) onSocket: (AsyncSocket *) *sock* wantsRunLoopForNewSocket:(AsyncSocket *) *newSocket* [optional]

Called when a new socket is spawned to handle a connection.

This method should return the run-loop of the thread on which the new socket and its delegate should operate. If omitted, [NSRunLoop currentRunLoop] is used.

10.2.1.10 - (void) onSocket: (AsyncSocket *) *sock* willDisconnectWithError:(NSError *) *err* [optional]

In the event of an error, the socket is closed.

You may call "unreadData" during this call-back to get the last bit of data off the socket. When connecting, this delegate method may be called before"onSocket:didAcceptNewSocket:" or "onSocket:didConnectToHost:".

10.2.1.11 - (void) onSocketDidDisconnect: (AsyncSocket *) *sock* [optional]

Called when a socket disconnects with or without error.

If you want to release a socket after it disconnects, do so here. It is not safe to do that during "onSocket:willDisconnectWithError:".

If you call the disconnect method, and the socket wasn't already disconnected, this delegate method will be called before the disconnect method returns.

10.2.1.12 - (void) onSocketDidSecure: (AsyncSocket *) *sock* [optional]

Called after the socket has successfully completed SSL/TLS negotiation.

This method is not called unless you use the provided startTLS method.

If a SSL/TLS negotiation fails (invalid certificate, etc) then the socket will immediately close, and the onSocket:willDisconnectWithError: delegate method will be called with the specific SSL error code.

10.2.1.13 - (BOOL) onSocketWillConnect: (AsyncSocket *) *sock* [optional]

Called when a socket is about to connect.

This method should return YES to continue, or NO to abort. If aborted, will result in AsyncSocketCanceledError.

If the connectToHost:onPort:error: method was called, the delegate will be able to access and configure the CFReadStream and CFWriteStream as desired prior to connection.

If the connectToAddress:error: method was called, the delegate will be able to access and configure the CFSocket and CFSocketNativeHandle (BSD socket) as desired prior to connection. You will be able to access and configure the CFReadStream and CFWriteStream in the onSocket:didConnectToHost:port: method.

The documentation for this protocol was generated from the following file:

- AsyncSocket.h

10.3 CMD_DEF Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- unsigned short int length
- unsigned char protoVer
- unsigned char command

10.3.1 Member Data Documentation

10.3.1.1 unsigned char CMD_DEF::command

10.3.1.2 unsigned short int CMD_DEF::length

10.3.1.3 unsigned char CMD_DEF::protoVer

The documentation for this struct was generated from the following file:

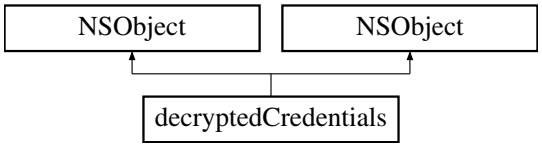
- Barcode_host.h

10.4 decryptedCredentials Class Reference

Encapsulating data class utilized by VFIConnect.

```
#import <decryptedCredentials.h>
```

Inheritance diagram for decryptedCredentials:



Instance Methods

- (void) - clear

  clears all decryptedCredentials properties
- (void) - clear

  clears all decryptedCredentials properties

Properties

- NSString ∗ clientID

  Client ID.
- NSString ∗ deviceKey

  Device Key.
- NSString ∗ serialNumber

  Serial Number.
- NSString ∗ deviceType

  Device Type.

10.4.1 Detailed Description

Encapsulating data class utilized by VFIConnect.

10.4.2 Method Documentation

10.4.2.1 - (void) clear

clears all decryptedCredentials properties

10.4.2.2 - (void) clear

clears all decryptedCredentials properties

10.4.3 Property Documentation

10.4.3.1 - (NSString ∗) clientID `[read],[write],[nonatomic],[retain]`

Client ID.

10.4.3.2 - (NSString *) deviceKey [read], [write], [nonatomic], [retain]

Device Key.

10.4.3.3 - (NSString *) deviceType [read], [write], [nonatomic], [retain]

Device Type.

10.4.3.4 - (NSString *) serialNumber [read], [write], [nonatomic], [retain]

Serial Number.

The documentation for this class was generated from the following files:

- decryptedCredentials.h
- VFIDecryptedCredentials.h

10.5 HOST_RX_BUF Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- unsigned long length
- unsigned long sequence
- unsigned long status

10.5.1 Member Data Documentation

10.5.1.1 unsigned long HOST_RX_BUF::length

10.5.1.2 unsigned long HOST_RX_BUF::sequence

10.5.1.3 unsigned long HOST_RX_BUF::status

The documentation for this struct was generated from the following file:

- Barcode_host.h

10.6 HOST_TX_BUF Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- unsigned long length
- unsigned long sequence

10.6.1 Member Data Documentation

10.6.1.1 unsigned long HOST_TX_BUF::length

10.6.1.2 unsigned long HOST_TX_BUF::sequence

The documentation for this struct was generated from the following file:

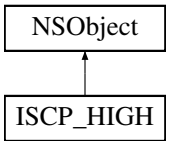
- Barcode_host.h

10.7 ISCP_HIGH Class Reference

Encapsulating data class utilized by VFIBarcode.

```
#import <ISCP_HIGH.h>
```

Inheritance diagram for ISCP_HIGH:



Instance Methods

- (void) - clear

  clears all ISCP_HIGH properties

Properties

- unsigned char commandType

  Type of command.
- unsigned char group

  setup/control/status/event
- unsigned char fid

  frame ID
- NSData * param

  data parameter(s)

10.7.1 Detailed Description

Encapsulating data class utilized by VFIBarcode.

10.7.2 Method Documentation

10.7.2.1 - (void) clear

clears all ISCP_HIGH properties

10.7.3 Property Documentation

10.7.3.1 - (unsigned char) commandType [read], [write], [atomic]

Type of command.

10.7.3.2 - (unsigned char) fid [read], [write], [atomic]

frame ID

10.7.3.3 - (unsigned char) group [read], [write], [atomic]

setup/control/status/event

10.7.3.4 - (NSData *) param [read], [write], [nonatomic], [retain]

data parameter(s)

The documentation for this class was generated from the following file:

- ISCP_HIGH.h

10.8 M_BEEP_DEF Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- int freq_1
- int duration_1
- int pause_1
- int freq_2
- int duration_2
- int pause_2
- int freq_3
- int duration_3
- int pause_3
- int freq_4
- int duration_4
- int pause_4
- int freq_5
- int duration_5
- int pause_5
- int freq_6
- int duration_6
- int pause_6

10.8.1 Member Data Documentation

10.8.1.1 int M_BEEP_DEF::duration_1

10.8.1.2 int M_BEEP_DEF::duration_2

10.8.1.3 int M_BEEP_DEF::duration_3

10.8.1.4 int M_BEEP_DEF::duration_4

10.8.1.5 int M_BEEP_DEF::duration_5

10.8.1.6 int M_BEEP_DEF::duration_6

10.8.1.7 int M_BEEP_DEF::freq_1

10.8.1.8 int M_BEEP_DEF::freq_2

10.8.1.9 int M_BEEP_DEF::freq_3

10.8.1.10 int M_BEEP_DEF::freq_4

10.8.1.11 int M_BEEP_DEF::freq_5

10.8.1.12 int M_BEEP_DEF::freq_6

10.8.1.13 int M_BEEP_DEF::pause_1

10.8.1.14 int M_BEEP_DEF::pause_2

10.8.1.15 int M_BEEP_DEF::pause_3

10.8.1.16 int M_BEEP_DEF::pause_4

10.8.1.17 int M_BEEP_DEF::pause_5

10.8.1.18 int M_BEEP_DEF::pause_6

The documentation for this struct was generated from the following file:

- Barcode_host.h

10.9 PARM_DEF Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- unsigned char length
- unsigned char pId

10.9.1 Member Data Documentation

10.9.1.1 unsigned char PARM_DEF::length

10.9.1.2 unsigned char PARM_DEF::pld

The documentation for this struct was generated from the following file:

- Barcode_host.h

10.10 POWER_STATUS Struct Reference

POWER_STATUS structure.

```
#import <VFIControl.h>
```

Protected Attributes

- bool HOST_PWR

  On Battery Power.
- bool EXT_PWR

  On External Power.
- bool POWERSHARE_ON

  Power Share Currently Active.
- bool DATA_SYNC_EN

  Data Sync is enabled.
- bool IDEVICE_PRESENT

  iDevice is connected to terminal
- bool GANG_CHARGER

  terminal on gang charger

10.10.1 Detailed Description

POWER_STATUS structure.

Structure to hold IAP Manager power status returned from getPowerStatus().

10.10.2 Member Data Documentation

10.10.2.1 - (bool) DATA_SYNC_EN [protected]

Data Sync is enabled.

10.10.2.2 - (bool) EXT_PWR [protected]

On External Power.

10.10.2.3 - (bool) GANG_CHARGER [protected]

terminal on gang charger

10.10.2.4 - (bool) HOST_PWR [protected]

On Battery Power.

10.10.2.5 - (bool) IDEVICE_PRESENT [protected]

iDevice is connected to terminal

10.10.2.6 - (bool) POWERSHARE_ON [protected]

Power Share Currently Active.

The documentation for this struct was generated from the following file:

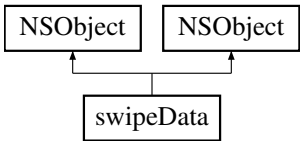
- VFIControl.h

10.11 swipeData Class Reference

Encapsulating data class utilized by VFIConnect.

`#import <swipeData.h>`

Inheritance diagram for swipeData:



Instance Methods

- (void) - clear

  clears all swipeData properties
- (void) - clear

  clears all swipeData properties

Properties

- NSString * maskedPAN

  Masked Card Account Number.
- CardType cardType

  Card Type.
- NSString * firstName

Account Holder First Name.

- NSString * lastName

  Account Holder Last Name.

10.11.1 Detailed Description

Encapsulating data class utilized by VFIConnect.

10.11.2 Method Documentation

10.11.2.1 - (void) clear

clears all swipeData properties

10.11.2.2 - (void) clear

clears all swipeData properties

10.11.3 Property Documentation

10.11.3.1 - (CardType) cardType [read], [write], [atomic]

Card Type.

10.11.3.2 - (NSString *) firstName [read], [write], [nonatomic], [retain]

Account Holder First Name.

10.11.3.3 - (NSString *) lastName [read], [write], [nonatomic], [retain]

Account Holder Last Name.

10.11.3.4 - (NSString *) maskedPAN [read], [write], [nonatomic], [retain]

Masked Card Account Number.

The documentation for this class was generated from the following files:

- swipeData.h
- VFISwipeData.h

10.12 symb_INFO Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- char codeID
- char AIMID
- unsigned short int symbology

10.12.1 Member Data Documentation

10.12.1.1 char symb_INFO::AIMID

10.12.1.2 char symb_INFO::codeID

10.12.1.3 unsigned short int symb_INFO::symbology

The documentation for this struct was generated from the following file:

- Barcode_host.h

10.13 T_BEEP Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- int freq
- int dur

10.13.1 Member Data Documentation

10.13.1.1 int T_BEEP::dur

10.13.1.2 int T_BEEP::freq

The documentation for this struct was generated from the following file:

- Barcode_host.h

10.14 T_BEEP_DEF Struct Reference

```
#include <Barcode_host.h>
```

Public Attributes

- T_BEEP b1
- int bPause
- T_BEEP b2

10.14.1 Member Data Documentation

10.14.1.1 T_BEEP T_BEEP_DEF::b1

10.14.1.2 T_BEEP T_BEEP_DEF::b2

10.14.1.3 int T_BEEP_DEF::bPause

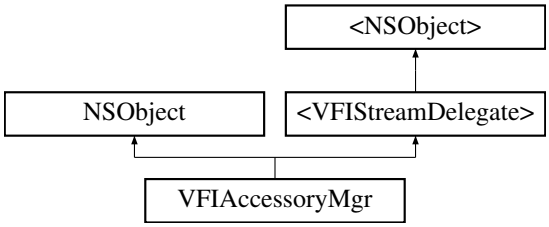
The documentation for this struct was generated from the following file:

- Barcode_host.h

10.15 VFIAccessoryMgr Class Reference

```
#import <VFIAccessoryMgr.h>
```

Inheritance diagram for VFIAccessoryMgr:



Instance Methods

- (void) - barcodeNotfication:
- (void) - controlNotfication:
- (void) - pinpadNotfication:
- (VFIStream ∗) - getXPI
- (VFIStream ∗) - getBarcode
- (VFIStream ∗) - getControl
- (VFIStream ∗) - getZontalk
- (VFIStream ∗) - getSled
- (VFIStream ∗) - getPrinter
- (void) - closeAccessory:
- (void) - closeDevices
- (void) - closeZontalk

Class Methods

- (VFIAccessoryMgr ∗) + sharedController

Protected Attributes

- NSMutableDictionary ∗ _sessions
- NSMutableArray ∗ _unusedProtocols
- BOOL _isBTDevice

Properties

- NSMutableDictionary * sessions
- NSMutableArray * unusedProtocols
- VFIStream * btChannel
- BOOL isBTDevice
- BOOL postClosing

10.15.1 Method Documentation

10.15.1.1 - (void) barcodeNotfication: (BOOL) *notify*

10.15.1.2 - (void) closeAccessory: (NSString *) *protocol*

10.15.1.3 - (void) closeDevices

10.15.1.4 - (void) closeZontalk

10.15.1.5 - (void) controlNotfication: (BOOL) *notify*

10.15.1.6 - (VFIStream*) getBarcode

10.15.1.7 - (VFIStream*) getControl

10.15.1.8 - (VFIStream*) getPrinter

10.15.1.9 - (VFIStream*) getSled

10.15.1.10 - (VFIStream*) getXPI

10.15.1.11 - (VFIStream*) getZontalk

10.15.1.12 - (void) pinpadNotfication: (BOOL) *notify*

10.15.1.13 + (VFIAccessoryMgr *) sharedController

10.15.2 Member Data Documentation

10.15.2.1 - (BOOL) _isBTDevice [protected]

10.15.2.2 - (NSMutableDictionary*) _sessions [protected]

10.15.2.3 - (NSMutableArray*) _unusedProtocols [protected]

10.15.3 Property Documentation

10.15.3.1 - (VFIStream*) btChannel [read], [nonatomic], [assign]

10.15.3.2 - (BOOL) isBTDevice [read], [atomic], [assign]

10.15.3.3 - (BOOL) postClosing [read], [atomic], [assign]

10.15.3.4 - (NSMutableDictionary*) sessions [read], [nonatomic], [assign]

10.15.3.5 - (NSMutableArray*) unusedProtocols [read], [nonatomic], [assign]

The documentation for this class was generated from the following file:

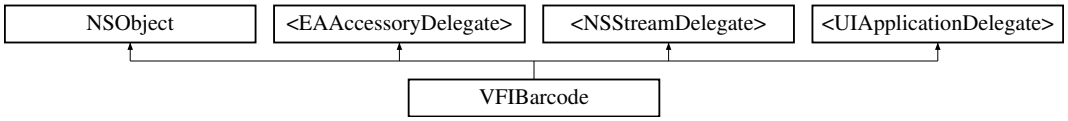
- VFIAccessoryMgr.h

10.16 VFIBarcode Class Reference

API methods for e210/e255/e315 Pinpad Barcode Scanner.

`#import <VFIBarcode.h>`

Inheritance diagram for VFIBarcode:



Instance Methods

- (id) - init

  Creates an instance of VFIBarcode class.
- (void) - initDevice

  Initializes the barcode device.
- (void) - initDeviceOnServer
- (void) - startScan

  Powers up the scanner engine.
- (void) - abortScan

  Powers down the scanner engine.
- (void) - sendISCP:

  Sends parameters to modify scanner properties.
- (void) - configureBeep:

  Sets the default beep properties.
- (void) - setBeep:

  DEPRECATED: Sets the default beep properties.
- (void) - sendBeep:

  Executes a barcode beep.
- (void) - setBeepOff

  Silences the barcode beep.
- (void) - setBeepOn

  Enables the barcode beep.
- (T_BEEP_DEF) - getBeep

  Retrieves the default barcode beep values.
- (void) - setScanTimeout:

  Sets barcode scanner timeout.

- (void) - setScanner2D

  Sets barcode scanner in 2D mode.
- (void) - setScanner1D

  Sets barcode scanner in 1D mode.
- (void) - setMultiScan

  Enables continuous scanning of barcodes.
- (void) - setSingleScan

  Disables continuous scanning of barcodes.
- (void) - getVersion

  Barcode application version.
- (void) - setLevel

  Sets Level Scanning Mode.
- (void) - setEdge

  Sets Edge Scanning Mode.
- (void) - setSoft

  Sets Software Scanning Mode.
- (void) - setPassive

  Sets Passive Scanning Mode.
- (void) - logEnabled:

  Enabled logging to delegate.
- (void) - consoleEnabled:

  Enabled logging to XCode Console.
- (void) - ignoreDuplicates:

  Duplicate barcode checking.
- (void) - clearDuplicatesBuffer

  Clears duplicate barcode checking buffer.
- (void) - mDeviceOpenForce
- (void) - sendTriggerEvent:

  Software control of scanner light.
- (void) - barcodeTypeEnabled:

  Enabled barcode type reporting.
- (void) - resetScannerToFactoryDefaults

  Reset Scanner.
- (void) - setScanButtonMode:

  Enable automatic trigger events in Soft Mode.
- (BOOL) - getScanButtonMode

  Returns scan button mode.
- (void) - includeAllBarcodeTypes

  Clears barcode type filtering look up table.
- (void) - includeBarcodeType:

  Removes a barcode type from the filtering look up table.
- (void) - excludeBarcodeType:

  Adds a barcode type from the filtering look up table.
- (void) - activateBarcodeType:

  Activates barcode type recognition in scanner.
- (void) - deactivateBarcodeType:

  Deactivates barcode type recognition in scanner.

- (void) - executeCmd:

  Allows direct stream access to Barcode Application.
- (void) - beepOnParsedScan:

  Controls beep execution after succssful scan.
- (void) - enableSoftTrigger:

  Ignore/Respond to trigger press events.
- (void) - closeDevice

  Terminates stream connection to Barcode.
- (void) - restartLoopDelay:

  Controls the restart loop delay.
- (void) - setInitLoop:

  Controls the amount of looping attempts to contact barcode application.
- (void) - setTrigger:

  Sets Barcode Trigger Mode.
- (void) - enableBlocking

  Renables default blocking on all API methods.
- (void) - disableBlocking

  Disables default blocking on all API methods.
- (NSString ∗) - frameworkVersion

  Reports current framework version.
- (NSString ∗) - scannerFirmwareVersion

  Reports current barcode Firmware version.
- (void) - processReceivedData:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - ignoreDisconnect

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - disableProtocol:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - sendData:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - sendCommand:data:

  Sends a e315 Barcode Command and Parameter.
- (void) - sendCommand:parameter:data:

  Sends a e315 Barcode Command and Parameter.
- (void) - sendCommand:parameter:

  Sends a e315 Barcode Command and Parameter.
- (void) - sendCommand:

  Sends a e315 Barcode Command.
- (void) - mDeviceOpen

  e315 Device Open.
- (void) - mDeviceClose

  e315 Device Close.
- (void) - mStartScan

  e315 Start Scan.
- (void) - mStopScan

  e315 Stop Scan.
- (void) - mTriggerMode:

Sets the e315 Scanning Mode.

- (void) - mBeepImmediate:

  Executes a e315 device beep.
- (NSString ∗) - mAppVersion

  Retrieves the e315 scanner application version.
- (NSString ∗) - mFirmwareVersion

  Retrieves the e315 scanner firmware version.
- (void) - mAutoBeepConfig:

  Enables or disables the e315 auto beep.
- (void) - mAutoBeepConfig:scanBeep:errorBeep:

  Enables or disables the e315 auto beep and configures the tone.
- (int) - mButtonStatus

  Reports the current state of the barcode trigger buttons on e315.
- (void) - mRestoreDefaults

  Rstores default settings on e315.
- (void) - mPicklistMode:

  Enables or disables Picklist Mode the e315.
- (void) - mScanTimeout:

  Sets the scan timeout the e315.
- (void) - mDifferentSymbolDecodesTimeout:

  Sets the multiple read timeout the e315.
- (void) - mContinuousMode:

  Enables or disables Continuous Mode the e315.
- (void) - mUniqueCodeReport:

  Enables or disables Unique Barcode Mode the e315.
- (void) - mElectronicDisplayRead:

  Enables or disables Electronic Display Read Mode the e315.
- (int) - mGetAutoBeepMode

  Retrieves the e315 auto beep mode.
- (T_BEEP_DEF) - mGetAutoBeepScan

  Retrieves the scan tone value of the current e315 auto beep mode.
- (T_BEEP_DEF) - mGetAutoBeepError

  Retrieves the error tone value of the current e315 auto beep mode.
- (void) - mScanDataPrefix:value:

  Sets the e315 scan data prefix.
- (void) - mScanDataSuffix1:value:

  Sets the e315 scan data suffix1.
- (void) - mScanDataSuffix2:value:

  Sets the e315 scan data suffix2.
- (void) - mScanDataTransmitFormat:

  Enables the e315 Scan Data Transmit Format.
- (void) - mAimPattern:

  Enables the e315 Aiming Pattern.
- (void) - mSymbology:value:

  Manage the e315 Symbology.
- (void) - mSymbology:data:

  Manage the e315 Symbology.

- (void) - mDisableAllSymbologies

  Disable e315 Symbologies.
- (int) - E08_RSA:publicKeyID:

  Load Key Information.
- (int) - loadKeyInformation_RSA:publicKeyID:

  Load Key Information.
- (int) - E08_x509:

  Load Key Information.
- (int) - loadKeyInformation_x509:

  Get PKI Ciphered Data.
- (int) - E12:p75Data:

  Load PKI Signature Files.
- (int) - loadPKISig:p75Data:

  Load PKI Signature Files.
- (int) - E10:

  Enable PKI Encryption.
- (int) - enablePKI:

  Enable PKI Encryption.

Class Methods

- (VFIBarcode *) + sharedController

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Properties

- id< VFIBarcodeDelegate > delegate

  Gets or Sets delegate for protocols.
- NSString * barcodeName

  Read Only name reported to External Accessory.
- NSString * barcodeManufacturer

  Read Only Manufacturer reported to External Accessory.
- NSString * barcodeModelNumber

  Read Only Model Number reported to External Accessory.
- NSString * barcodeSerialNumber

  Read Only Serial Number reported to External Accessory.
- NSString * barcodeFirmwareRevision

  Read Only Firmware Version reported to External Accessory.
- NSString * barcodeHardwareRevision

  Read Only Hardware Version reported to External Accessory.
- NSString * barcodeVersion

  Read Only Barcode App version set by getVersion
- NSString * firmware

  Read Only Scanner firmware set by scannerFirmwareVersion
- BOOL barcodeConnected

  Read Only Boolean barcode connection status.

- BOOL connected

  Read Only Boolean barcode connection status.
- EAAccessory * eaACC

  Read Only Connected Accessory.
- BOOL BTconnected

  Read Only Boolean barcode e210/e255/e315 Bluetooth connection Status.
- BOOL initialized

  Read Only Boolean barcode initialized.
- BOOL isGen3

  Read Only Boolean Gen3 Connected.
- NSString * keyLoadErrorCode

  Error code returned from unsuccessful Key Loading.

10.16.1 Detailed Description

API methods for e210/e255/e315 Pinpad Barcode Scanner.

Implementing this class will allow API calls that will access e210/e255/e315 barcode hardware (if available).

10.16.2 Method Documentation

10.16.2.1 - (void) abortScan

Powers down the scanner engine.

Executing abortScan will remove power from the scanning engine deactivating barcode scanning.

10.16.2.2 - (void) activateBarcodeType: (int) *type*

Activates barcode type recognition in scanner.

Parameters

type	The barcode to ACTIVATE RECOGNITION in the scanner. Value is from barcodeTypes

This method sends appropriate ISCP command to the scanner to turn on the recognition of specific barcode types.

10.16.2.3 - (void) barcodeTypeEnabled: (BOOL) *enable*

Enabled barcode type reporting.

Parameters

enable	Setting *TRUE* enables barcode types to be sent with barcode scan data to VFIBarcodeDelegate-::barcodeScanData:thetype:()

10.16.2.4 - (void) beepOnParsedScan: (BOOL) *enable*

Controls beep execution after succssful scan.

Parameters

enable	A value of *TRUE* will allow a beep to be played after a successful barcode capture, while *FALSE* will supress the beep.

Barcode beep is ON by default.

10.16.2.5 - (void) clearDuplicatesBuffer

Clears duplicate barcode checking buffer.

Calling this method will reset the duplicate barcode checking buffer.

10.16.2.6 - (void) closeDevice

Terminates stream connection to Barcode.

This method will shut down the connection to the Barcode stream. An initDevice will need to be executed again to engage a new stream connection.

10.16.2.7 - (void) configureBeep: (T_BEEP_DEF) *beep*

Sets the default beep properties.

Parameters

beep	T_BEEP_DEF packet containing Beep1, pause, and Beep2

This method will configure the default beep properties. This value does not persist between power resets. This method replaces setBeep:

Example Usage:

```
T_BEEP_DEF b;
T_BEEP b1;
T_BEEP b2;

b1.freq = 0x40;
b1.dur = 0x30;
b2.freq = 0x35;
b2.dur = 0x30;

b.b1 = b1;
b.b2 = b2;
b.bPause = 0x50;

[barcode configureBeep:b];
```

10.16.2.8 - (void) consoleEnabled: (BOOL) *enable*

Enabled logging to XCode Console.

Parameters

enable	Setting *TRUE* enables additional logging to debug b_console window in iOS

10.16.2.9 - (void) deactivateBarcodeType: (int) *type*

Deactivates barcode type recognition in scanner.

Parameters

type	The barcode to DEACTIVATE RECOGNITION in the scanner. Value is from barcodeTypes

This method sends appropriate ISCP command to the scanner to turn off the recognition of specific barcode types.

10.16.2.10 - (void) disableBlocking

Disables default blocking on all API methods.

By default blocking is on. Most API calls will wait for a response from Barcode app before returning control to integrator. This behavior can be turned off by calling this method. Default blocking can be turned back on by calling enableBlocking

10.16.2.11 - (void) disableProtocol: (BOOL) *disable*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.16.2.12 - (int) E08_RSA: (NSString ∗) *publicKeyRSA* publicKeyID:(NSString ∗) *keyID*

Load Key Information.

See Also

- loadKeyInformation_RSA:publicKeyID:

10.16.2.13 - (int) E08_x509: (NSString ∗) *publicKeyX509*

Load Key Information.

See Also

- loadKeyInformation_x509:

10.16.2.14 - (int) E10: (BOOL) *enable*

Enable PKI Encryption.

See Also

- enablePKI:

10.16.2.15 - (int) E12: (int) *type* p75Data:(NSData ∗) *data*

Load PKI Signature Files.

See Also

- loadPKISig:p75Data:

10.16.2.16 - (void) enableBlocking

Renables default blocking on all API methods.

By default blocking is on. Most API calls will wait for a response from Barcode app before returning control to integrator. This behavior can be turned off by calling disableBlocking

10.16.2.17 - (int) enablePKI: (bool) *enable*

Enable PKI Encryption.

This command will be used to enable PKI scan encryption

XPI reference: E10

Parameters

enable	TRUE = enable scan encryption:

Return values

Result	Code: • 00: PKI Mode Set. • 01: No Encryption.

10.16.2.18 - (void) enableSoftTrigger: (BOOL) *enable*

Ignore/Respond to trigger press events.

Parameters

enable	A value of *TRUE* will respond to trigger press events during Soft Mode. A value of *FALSE* will ignore trigger press events

This convenience funtion allows the integrator to stay in Soft Mode and be able to ignore trigger presses without having to shut down power to scanner with an abortScan.

10.16.2.19 - (void) excludeBarcodeType: (int) *type*

Adds a barcode type from the filtering look up table.

Parameters

type	The barcode to INCLUDE from the filtering look up table. Value is from barcodeTypes

This method will filter out the desired barcode type.

10.16.2.20 - (void) executeCmd: (NSData ∗) *data*

Allows direct stream access to Barcode Application.

Parameters

data	The data to send to the barcode application

This method will pass the provided NSData object directly to the barcode input stream.

10.16.2.21 - (NSString∗) frameworkVersion

Reports current framework version.

10.16.2.22 - (T_BEEP_DEF) getBeep

Retrieves the default barcode beep values.

Return values

getBeep	T_BEEP_DEF structure that contains Beep1, Beep2, and the pause between them.

This method polls the barcode for the current default beep values.

10.16.2.23 - (BOOL) getScanButtonMode

Returns scan button mode.

Return values

BOOL	Current scan button mode that the framework is operating in.

10.16.2.24 - (void) getVersion

Barcode application version.

After the successful execution of this blocking method, the barcode application version can be retrieved by polling the property barcodeVersion.

10.16.2.25 - (void) ignoreDisconnect

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.16.2.26 - (void) ignoreDuplicates: (BOOL) *enable*

Duplicate barcode checking.

Parameters

enable	Setting *TRUE* will filter out any duplicte barcodes that have been scanned since this value was set.

10.16.2.27 - (void) includeAllBarcodeTypes

Clears barcode type filtering look up table.

When barcode data is received, if there are any entries in the filter look up table from executing excludeBarcodeType:, the table is scanned for a match to the current barcode data, and if match is found, that barcode is NOT passed to the integrator. This method clears out the filter look up table so no filtering will take place while the filtering look up table has 0 entries in it.

10.16.2.28 - (void) includeBarcodeType: (int) *type*

Removes a barcode type from the filtering look up table.

Parameters

type	The barcode to EXCLUDE from the filtering look up table. Value is from barcodeTypes

This method will insure that the desired barcode will not be filtered by any previous table insertions of the desired barcode type into the filtering look up table.

10.16.2.29 - (id) init

Creates an instance of VFIBarcode class.

Return values

<*id*>	of VFIBarcode class

Example Usage:

```
VFIBarcode* barcode = [[VFIBarcode alloc] init];
```

10.16.2.30 - (void) initDevice

Initializes the barcode device.

This is executed after the instance is created with init. If any of the optional protocols will be used, setDelegate should first be executed.

Example Usage:

```
VFIBarcode* barcode = [[VFIBarcode alloc] init];
[barcode setDelegate:self];
[barcode initDevice];
```

10.16.2.31 - (void) initDeviceOnServer

10.16.2.32 - (int) loadKeyInformation_RSA: (NSString ∗) *publicKeyRSA* publicKeyID:(NSString ∗) *keyID*

Load Key Information.

The E08 command can be used to load the initial PKI Public Key, or update the existing key. encryption.

Upon unsuccessful response, error code can be retrieved in keyLoadErrorCode

XPI reference: E08

Parameters

publicKeyRSA	RSA Public Key in Base 64 format
keyID	Key Identifer

Return values

Result	Code: • 00: Successful. • 01: Failure. Refer to keyLoadErrorCode

10.16.2.33 - (int) loadKeyInformation_x509: (NSString ∗) *publicKeyX509*

Get PKI Ciphered Data.

The E08 command can be used to load the initial PKI Public Key, or update the existing key.

Upon unsuccessful response, error code can be retrieved in keyLoadErrorCode

XPI reference: E08

Parameters

publicKeyX509	X.509 Public Key Cert in Base 64

Return values

Result	Code: • 00: Successful. • 01: Failure. Refer to keyLoadErrorCode

10.16.2.34 - (int) loadPKISig: (int) *type* p75Data:(NSData ∗) *data*

Load PKI Signature Files.

This command will be used to inject the P7S binary data associated to the “PKI Public Key File” and “Key ID File”. This command must be executed prior to the execution of E08 command.

XPI reference: E12

Parameters

type	Sign File Data associated to: • 01: P7S Data is for PKI Public Key File. • 02: P7S Data is for Key ID File

data	Hex data of the P7S file associated with the Public Key or Key ID Files

Return values

Result	Code: • 00: Successful. • 01: Failure.

10.16.2.35 - (void) logEnabled: (BOOL) *enable*

Enabled logging to delegate.

Parameters

enable	Setting *TRUE* enables logging to barcodeLogEntry:withSeverity: (VFIBarcodeDelegate-p)

10.16.2.36 - (void) mAimPattern: (BOOL) *isActive*

Enables the e315 Aiming Pattern.

Enables or disables AIM pattern. When enabled, the cross-hair pattern for the laser is used during bar code capture.

Parameters

isActive	True = ON, False = OFF

10.16.2.37 - (NSString∗) mAppVersion

Retrieves the e315 scanner application version.

Return values

mAppVersion	Returns a 6 character NSString version ID for the e315 reader.

10.16.2.38 - (void) mAutoBeepConfig: (int) *mode*

Enables or disables the e315 auto beep.

Parameters

mode	Integer value representing mode

- 0 = auto beep disabled
- 1 = auto beep enabled, scan beeps only
- 2 = auto beep enabled, error beeps only
- 3 = auto beep enabled, scan and error beeps

By default, Continuous mode is enabled. If Continuous mode is disabled, a single scan mode is enabled, where this mode scan session is turned off after one successful barcode scan or when the timeout period expires.

NOTE: Default auto beep configuration settings are enforced at terminal power cycle and restart.

10.16.2.39 - (void) mAutoBeepConfig: (int) *mode* scanBeep:(T_BEEP_DEF) *scan* errorBeep:(T_BEEP_DEF) *error*

Enables or disables the e315 auto beep and configures the tone.

Parameters

mode	Integer value representing mode
scan	T_BEEP_DEF structure representing the default scan beep tone
error	T_BEEP_DEF structure representing the default error beep tone

- 0 = auto beep disabled
- 1 = auto beep enabled, scan beeps only, configure scan beep tone
- 2 = auto beep enabled, error beeps only,configure error beep tone
- 3 = auto beep enabled, scan and error beeps, configure scan and error beep tones

By default, Continuous mode is enabled. If Continuous mode is disabled, a single scan mode is enabled, where this mode scan session is turned off after one successful barcode scan or when the timeout period expires.

NOTE: Default auto beep configuration settings are enforced at terminal power cycle and restart.

10.16.2.40 - (void) mBeepImmediate: (M_BEEP_DEF) *beep*

Executes a e315 device beep.

This method will execute a beep from the e315 scanner. Duration values of 0 signify no beep

Parameters

beep	M_BEEP_DEF packet containing values for up to 6 beeps to be executed

Example Usage:

```
M_BEEP_DEF b;

b.freq_1 = 0x40;
b.duration_1 = 0x30;
b.pause_1 = 0x50;

b.freq_2 = 0x35;
b.duration_2 = 0x30;
b.pause_2 = 0x50;


[barcode mBeepImmediate:b];
```

10.16.2.41 - (int) mButtonStatus

Reports the current state of the barcode trigger buttons on e315.

Sends trigger status to the delegate barcodeTriggerEvent: (VFIBarcodeDelegate-p)

- 0: No Trigger Pressed

- 1: Left Trigger Pressed
- 2: Right Trigger Pressed
- 3: Both Triggers Pressed

10.16.2.42 - (void) mContinuousMode: (BOOL) *isActive*

Enables or disables Continuous Mode the e315.

Continuous mode is where the barcode laser scans continuously. Normally, the laser shuts off and the scanning session ends after one successful scan/decode. Continuous mode is enabled with any of the Trigger modes. By default, Continuous mode is enabled.

Parameters

isActive	True = ON, False = OFF

10.16.2.43 - (void) mDeviceClose

e315 Device Close.

Powers down the e315 and closes the communication port.

NOTE:This command should be executed upon program termination to avoid exessive powerage usage.

10.16.2.44 - (void) mDeviceOpen

e315 Device Open.

Powers e315 and opens the communication port for communication.

NOTE:This command must be sent before sending any other command.

10.16.2.45 - (void) mDeviceOpenForce

10.16.2.46 - (void) mDifferentSymbolDecodesTimeout: (float) *seconds*

Sets the multiple read timeout the e315.

In Continuous mode, use this method to prevent multiple reads of a symbol left in the decoder's field of view. The timeout begins when a symbol is removed from the field of view. The default value is 0.6 seconds.

Parameters

seconds	Timeout value, in seconds

10.16.2.47 - (void) mDisableAllSymbologies

Disable e315 Symbologies.

Disables all the e315 Symbologies

10.16.2.48 - (void) mElectronicDisplayRead: (BOOL) *isActive*

Enables or disables Electronic Display Read Mode the e315.

This method improves barcode reading performance with target bar codes displayed on mobile phones and electronic displays.

Parameters

isActive	True = ON, False = OFF

10.16.2.49 - (NSString∗) mFirmwareVersion

Retrieves the e315 scanner firmware version.

Return values

mFirmwareVersion	Returns the firmware version number/string(s) of the e315 scanner.

10.16.2.50 - (T_BEEP_DEF) mGetAutoBeepError

Retrieves the error tone value of the current e315 auto beep mode.

Return values

mGetAutoBeepError	T_BEEP_DEF value representing default error tone

NOTE: You must be in Mode 2 or Mode 3 to be able to retrieve the error value. See mGetAutoBeepMode

10.16.2.51 - (int) mGetAutoBeepMode

Retrieves the e315 auto beep mode.

Return values

mGetAutoBeepMode	Integer value representing mode

- 0 = auto beep disabled
- 1 = auto beep enabled, scan beeps only
- 2 = auto beep enabled, error beeps only
- 3 = auto beep enabled, scan and error beeps

By default, Continuous mode is enabled. If Continuous mode is disabled, a single scan mode is enabled, where this mode scan session is turned off after one successful barcode scan or when the timeout period expires.

NOTE: Default auto beep configuration settings are enforced at terminal power cycle and restart.

10.16.2.52 - (T_BEEP_DEF) mGetAutoBeepScan

Retrieves the scan tone value of the current e315 auto beep mode.

Return values

mGetAutoBeepScan	T_BEEP_DEF value representing default scan tone

NOTE: You must be in Mode 1 or Mode 3 to be able to retrieve the scan value. See mGetAutoBeepMode

10.16.2.53 - (void) mPicklistMode: (BOOL) *isActive*

Enables or disables Picklist Mode the e315.

Picklist mode is where the barcode reader decodes only bar codes that are aligned under the center of the aiming pattern.

Parameters

isActive	True = ON, False = OFF

10.16.2.54 - (void) mRestoreDefaults

Rstores default settings on e315.

Sets all parameters to default values

10.16.2.55 - (void) mScanDataPrefix: (BOOL) *isActive* value:(char) *b*

Sets the e315 scan data prefix.

Use this API when scan data transmission format prefix + data is selected to set the prefix value.

@param isActive True = ON, False = OFF

Parameters

b	Value 0x00 - 0xFF representing the prefix value

NOTE: Use mScanDataTransmitFormat: to enable/disable format

10.16.2.56 - (void) mScanDataSuffix1: (BOOL) *isActive* value:(char) *b*

Sets the e315 scan data suffix1.

Use this API when scan data transmission format data + suffix1 is selected to set the suffix1 value.

@param isActive True = ON, False = OFF

Parameters

b	Value 0x00 - 0xFF representing the suffix1 value

NOTE: Use mScanDataTransmitFormat: to enable/disable format

10.16.2.57 - (void) mScanDataSuffix2: (BOOL) *isActive* value:(char) *b*

Sets the e315 scan data suffix2.

Use this API when scan data transmission format data + suffix1 is selected to set the suffix2 value.

@param isActive True = ON, False = OFF

Parameters

b	Value 0x00 - 0xFF representing the suffix2 value

NOTE: Use mScanDataTransmitFormat: to enable/disable format

10.16.2.58 - (void) mScanDataTransmitFormat: (BOOL) *isActive*

Enables the e315 Scan Data Transmit Format.

Changes the scan data format to have a custom prefix and/or suffix1 and/or suffix2 value. The default is scan data as is.

Parameters

isActive	True = ON, False = OFF

10.16.2.59 - (void) mScanTimeout: (int) *seconds*

Sets the scan timeout the e315.

The timeout value range is 1 second to 255 seconds. In Continuous mode, the maximum allowed timeout is 255 seconds. In Single Scan Mode, the maximum allowed timeout value is 10 seconds.

Parameters

seconds	Timeout value, in seconds

10.16.2.60 - (void) mStartScan

e315 Start Scan.

Acatives the e315 scan engine. Triggers are monitored (except Passive mode)and events are executed depending on trigger mode.

Please see mTriggerMode: for operational characteristics.

NOTE:This command should be executed upon program termination to avoid exessive powerage usage.

10.16.2.61 - (void) mStopScan

e315 Stop Scan.

Decativates the e315 scan engine. Triggers are no longer monitored.

NOTE:The scanner is still in an "on" state and listening for commands. Use mDeviceClose to shut down power.

10.16.2.62 - (void) mSymbology: (int) *parameter* data:(NSData ∗) *value*

Manage the e315 Symbology.

Configures any supported symbology, including enabling or disabling a symbology and setting symbology parameters (for example, to set length, format, and supplements of a symbology). Refer to barcode_host.h file

Parameters

parameter	Parameter from barcode_host.h file
value	NSData value specified from barcode_host.h file

Example: To set the length range of C93 of 4 - 9 characters, the parameter would be SYM_PID_SETLEN_RANGE_C93 and the NSData value would be 0x0409

10.16.2.63 - (void) mSymbology: (int) *parameter* value:(int) *value*

Manage the e315 Symbology.

Configures any supported symbology, including enabling or disabling a symbology and setting symbology parameters (for example, to set length, format, and supplements of a symbology). Refer to barcode_host.h file

Parameters

parameter	Parameter from barcode_host.h file
value	Single int value specified from barcode_host.h file

Example: To transmit a UPCA check digit, the parameter would be SYM_PID_XMIT_UPCA_CHECK_DIGIT and the value would be 1

10.16.2.64 - (void) mTriggerMode: (int) *mode*

Sets the e315 Scanning Mode.

Parameters

mode	Integer value representing mode

- 0 = The EDGE parameter enables Edge Trigger mode, where the scan session starts when the barcode button(s) is pressed and released, and stays on until the timeout period expires or the barcode button(s) is pressed again and released.
- 1 = The LEVEL parameter enables Level Trigger mode, where the scan session starts when the barcode button(s) is pressed, and stays on until the button is released. The scan session turns off automatically when the specified timeout period is reached.
- 2 = The SOFT parameter enables Soft Trigger mode, where the host device issues a command to start the scan session, and the barcode button(s) have no control during the scan session.
- 3 = The PASSIVE parameter enables Passive Trigger mode that behaves the same as Soft Trigger mode, except that a BUTTON_STATUS command is not initiated. Button status is obtained by the host using the mButtonStatus command.

  By default, Continuous mode is enabled. If Continuous mode is disabled, a single scan mode is enabled, where this mode scan session is turned off after one successful barcode scan or when the timeout period expires.

  NOTE: The terminal reverts to the default Level mode at every terminal power cycle and restart.

10.16.2.65 - (void) mUniqueCodeReport: (BOOL) *isActive*

Enables or disables Unique Barcode Mode the e315.

While in continuous mode, use this method to only report unique bar codes during the scanning session.

Parameters

isActive	True = ON, False = OFF

10.16.2.66 - (void) processReceivedData: (NSData ∗) *data*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.16.2.67 - (void) resetScannerToFactoryDefaults

Reset Scanner.

Resets scanner engine to factory defaults.

10.16.2.68 - (void) restartLoopDelay: (float) *sec*

Controls the restart loop delay.

Parameters

sec	The amount of time in fractional sections to wait between attempts in establishing contact to e210/e255/e315 while waiting for initialization

Default is 1.0 seconds.

10.16.2.69 - (NSString∗) scannerFirmwareVersion

Reports current barcode Firmware version.

10.16.2.70 - (void) sendBeep: (T_BEEP_DEF) *beep*

Executes a barcode beep.

Parameters

beep	T_BEEP_DEF packet containing Beep1, pause, and Beep2

This method will execute a beep from the barcode scanner.

Example Usage:

```
T_BEEP_DEF b;
T_BEEP b1;
T_BEEP b2;

b1.freq = 0x40;
b1.dur = 0x30;
b2.freq = 0x35;
b2.dur = 0x30;

b.b1 = b1;
b.b2 = b2;
b.bPause = 0x50;

[barcode sendBeep:b];
```

10.16.2.71 - (void) sendCommand: (int) *command*

Sends a e315 Barcode Command.

Parameters

command	Command from Barcode_hosts.h

Please reference the Barcode_hosts.h file for command definitions.

10.16.2.72 - (void) sendCommand: (int) *command* data:(NSData ∗) *addData*

Sends a e315 Barcode Command and Parameter.

Parameters

command	Command from Barcode_hosts.h
addData	additional data needed for parameter requirements

Please reference the Barcode_hosts.h file for command/parameter definitions.

10.16.2.73 - (void) sendCommand: (int) *command* parameter:(int) *parameter*

Sends a e315 Barcode Command and Parameter.

Parameters

command	Command from Barcode_hosts.h
parameter	Parameter from Barcode_hosts.h

Please reference the Barcode_hosts.h file for command/parameter definitions.

10.16.2.74 - (void) sendCommand: (int) *command* parameter:(int) *parameter* data:(NSData ∗) *addData*

Sends a e315 Barcode Command and Parameter.

Parameters

command	Command from Barcode_hosts.h
parameter	Parameter from Barcode_hosts.h
addData	additional data needed for parameter requirements

Please reference the Barcode_hosts.h file for command/parameter definitions.

10.16.2.75 - (void) sendData: (NSData ∗) *data*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.16.2.76 - (void) sendISCP: (ISCP_HIGH ∗) *frame*

Sends parameters to modify scanner properties.

Parameters

frame	ISCP_HIGH packet containing command frameType, group, FID,and parameters

Creating ISCP_HIGH packets and sending them through this method sets various scanner properties.

Example Usage: To set the ability for the barcode scanner to recognize QR Codes, you must send frameType *TYPE_-SETUP_WRITE*, group 0x4b, fid 0x5a and parameters 0x00

```
ISCP_HIGH *frame = [[ISCP_HIGH alloc] init];
frame.commandType = TYPE_SETUP_WRITE;
frame.group = 0x4a;
frame.fid = 0x40;
unsigned char bytes[1] = { 0x01 };
frame.param = [[NSData alloc] initWithBytes:bytes length:1];
[[e210/e255/e315 barcode] sendISCP:frame];
[frame.param release];
[frame release];
```

10.16.2.77 - (void) sendTriggerEvent: (BOOL) *activate*

Software control of scanner light.

Parameters

activate	While is Soft Mode setSoft or Passive Mode setPassive, setting *TRUE* will enabable the scanner light, while *false* will deactivate it.

10.16.2.78 - (void) setBeep: (T_BEEP_DEF) *beep*

DEPRECATED: Sets the default beep properties.

Parameters

beep	T_BEEP_DEF packet containing Beep1, pause, and Beep2

Deprecated Please see configureBeep:

10.16.2.79 - (void) setBeepOff

Silences the barcode beep.

This method will silence the beep that is executed upon a successful barcode scan.

10.16.2.80 - (void) setBeepOn

Enables the barcode beep.

This method will allow the beep that is executed upon a successful barcode scan.

10.16.2.81 - (void) setEdge

Sets Edge Scanning Mode.

A reading session begins (lighting and decode processing on) when the Trigger line is activated. The reading session stops when an abortScan is executed, barcode data is received, or a timeout setScanTimeout: is reached. This differs from Level Mode setLevel in the fact that you do not need to continue to depress the trigger to keep the lighting and decode processing on. The light remains on after you release the trigger.

10.16.2.82 - (void) setInitLoop: (int) *loop*

Controls the amount of looping attempts to contact barcode application.

Parameters

loop	The number of loops in establishing contact to e210/e255/e315 while waiting for initialization

If the e210/e255/e315 is unresponsive, the framework will loop the specified number of times, with a delay of restartLoopDelay: between each attempt. The default is 59 loops.

10.16.2.83 - (void) setLevel

Sets Level Scanning Mode.

A reading session begins (lighting and decode processing on) when the Trigger line is activated. The reading session stops when an abortScan is executed, barcode data is received, trigger line deactivated, or a timeout setScanTimeout: is reached. This differs from Edge Mode setEdge in the fact that you need to continue to depress the trigger to keep the lighting and decode processing on.

10.16.2.84 - (void) setMultiScan

Enables continuous scanning of barcodes.

While in Level Mode setLevel, after a successful scan, the scanning light will remain on ready to capture the next barcode.

10.16.2.85 - (void) setPassive

Sets Passive Scanning Mode.

A reading session begins (lighting and decode processing on) when sendTriggerEvent: *TRUE* is sent. The reading session stops when an abortScan is executed, barcode data is received, sendTriggerEvent: *FALSE* is sent or a timeout setScanTimeout: is reached. Pressing the trigger does not activate the light. Trigger press/release events does not send any data to the delegate barcodeTriggerEvent: (VFIBarcodeDelegate-p) This differs from Soft Mode setSoft in the fact that you do not get any trigger press events in Passive Mode.

10.16.2.86 - (void) setScanButtonMode: (BOOL) *enable*

Enable automatic trigger events in Soft Mode.

Parameters

enable	Setting *TRUE* enables automatic sending of sendTriggerEvent: when barcodeTriggerEvent: (VFIBarcodeDelegate-p) receives a trigger event.

10.16.2.87 - (void) setScanner1D

Sets barcode scanner in 1D mode.

Changes the scanning characteristics to read barcodes that are linear and one dimensional.

10.16.2.88 - (void) setScanner2D

Sets barcode scanner in 2D mode.

Changes the scanning characteristics to read barcodes that two dimensional.

10.16.2.89 - (void) setScanTimeout: (long) *milliseconds*

Sets barcode scanner timeout.

Parameters

milliseconds	Interval that must pass in milliseconds before timeout error.

While in Edge mode setEdge or Level mode setLevel, clicking the trigger with the scanner engine ON will power the scanner light and remain on until a successful scan or the timeout threshold is reached that is set by this method. Timeout does not apply to Soft Mode setSoft or Passive Mode setPassive.

10.16.2.90 - (void) setSingleScan

Disables continuous scanning of barcodes.

While in Edge Mode setEdge or Level Mode setLevel, after a successful scan, the scanning light will turn off. This is default behavior.

10.16.2.91 - (void) setSoft

Sets Software Scanning Mode.

A reading session begins (lighting and decode processing on) when sendTriggerEvent: *TRUE* is sent. The reading session stops when an abortScan is executed, barcode data is received, sendTriggerEvent: *FALSE* is sent or a timeout setScanTimeout: is reached. Pressing the trigger does not activate the light. Trigger press/release events sends data to the delegate barcodeTriggerEvent: (VFIBarcodeDelegate-p) This differs from Passive Mode setPassive in the fact that you do not get any trigger press events in Passive Mode.

10.16.2.92 - (void) setTrigger: (int) *mode*

Sets Barcode Trigger Mode.

Parameters

mode	Barcode trigger mode, selected from scanningModes

This one method can have the same outcome as calling any one of the methods setSoft, setPassive, setLevel, setEdge.

10.16.2.93 + (VFIBarcode ∗) sharedController

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.16.2.94 - (void) startScan

Powers up the scanner engine.

Executing startScan will allow barcode scanning to commence.

10.16.3 Property Documentation

10.16.3.1 - (BOOL) barcodeConnected [read], [atomic], [assign]

Read Only Boolean barcode connection status.

10.16.3.2 - (NSString*) barcodeFirmwareRevision [read], [nonatomic], [assign]

Read Only Firmware Version reported to External Accessory.

10.16.3.3 - (NSString*) barcodeHardwareRevision [read], [nonatomic], [assign]

Read Only Hardware Version reported to External Accessory.

10.16.3.4 - (NSString*) barcodeManufacturer [read], [nonatomic], [assign]

Read Only Manufacturer reported to External Accessory.

10.16.3.5 - (NSString*) barcodeModelNumber [read], [nonatomic], [assign]

Read Only Model Number reported to External Accessory.

10.16.3.6 - (NSString*) barcodeName [read], [nonatomic], [assign]

Read Only name reported to External Accessory.

10.16.3.7 - (NSString*) barcodeSerialNumber [read], [nonatomic], [assign]

Read Only Serial Number reported to External Accessory.

10.16.3.8 - (NSString*) barcodeVersion [read], [nonatomic], [assign]

Read Only Barcode App version set by getVersion

10.16.3.9 - (BOOL) BTconnected [read], [atomic], [assign]

Read Only Boolean barcode e210/e255/e315 Bluetooth connection Status.

10.16.3.10 - (BOOL) connected [read], [atomic], [assign]

Read Only Boolean barcode connection status.

10.16.3.11 - (id< VFIBarcodeDelegate >) delegate [read], [write], [atomic], [retain]

Gets or Sets delegate for protocols.

10.16.3.12 - (EAAccessory*) eaACC [read], [nonatomic], [assign]

Read Only Connected Accessory.

10.16.3.13 - (NSString*) firmware [read], [nonatomic], [assign]

Read Only Scanner firmware set by scannerFirmwareVersion

10.16.3.14 - (BOOL) initialized [read], [atomic], [assign]

Read Only Boolean barcode initialized.

10.16.3.15 - (BOOL) isGen3 [read], [atomic], [assign]

Read Only Boolean Gen3 Connected.

10.16.3.16 - (NSString*) keyLoadErrorCode [read], [nonatomic], [assign]

Error code returned from unsuccessful Key Loading.

Return Values:

- 00000000: Successful PKI update.
- 00001000: Parameter Error. NULL or out of Range.
- 00001001: Invalid Public Key File.
- 00001002: Encryption Failed.
- 00001003: Too small output buffer.
- 00001004: Invalid Key ID File.
- 00001005: Error in opening DataFile.
- 00001006: Error in Hash Key computation.
- 00001100: Unknown or unexpected error.

The documentation for this class was generated from the following file:

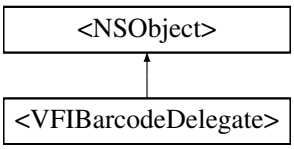
- VFIBarcode.h

10.17 <VFIBarcodeDelegate> Protocol Reference

Protocol methods established for VFIBarcode class.

`#import <VFIBarcode.h>`

Inheritance diagram for <VFIBarcodeDelegate>:



Instance Methods

- (void) - commandResult:

  Receives int value commResult.

- (void) - barcodeTriggerEvent:

  Receives int value triggerResponse.

- (void) - barcodeLogEntry:withSeverity:

  When logEnabled: (VFIBarcode) is passed `TRUE`, this delegate will receive log entries.

- (void) - barcodeSerialData:incoming:

  All incoming/outgoing data going to the Barcode application can be monitored through this delegate.

- (void) - barcodeInitialized:

  Notifies of initialization state changes in e210/e255/e315 Barcode Application.

- (void) - barcodeReconnectStarted

  When the External Accessory reports the barcode devices is detected (but not initialized by framework),this delegate is called. This signifies the beginning of the framework initialization process of the barcode application.

- (void) - barcodeReconnectFinished

  This signifies the end of the framework initialization process for barcode application.

- (void) - barcodeConnected:

  Notified of barcode connection/disconnection events.

- (void) - barcodeDataReceived:

  This delegate monitors all data received from Barcode Application.

- (void) - barcodeDataSent:

  This delegate monitors all data sent from Barcode Application.

- (void) - barcodeScanData:barcodeType:

  This delegate receives the barcode scan data.

- (void) - barcodeScanData:

  This delegate receives the encrypted barcode scan data.

- (void) - barcodeMessage:

  Additional messaging used by VMF.

- (void) - barcodeScanData:code:symbol:aim:

  This delegate receives the barcode scan data.

10.17.1 Detailed Description

Protocol methods established for VFIBarcode class.

10.17.2 Method Documentation

10.17.2.1 - (void) barcodeConnected: (BOOL) *isConnected* [optional]

Notified of barcode connection/disconnection events.

A connect/disconnect can either be from a physical disconnection/connection with the External Accessory API, or from an application going to backround or returning to foreground.

Parameters

isConnected	A new connection or disconnection was detected • `TRUE` The barcode application has connected. • `FALSE` The barcode application has disconnected.

10.17.2.2 - (void) barcodeDataReceived: (NSData *) *data* [optional]

This delegate monitors all data received from Barcode Application.

Parameters

data	A NSData binary object representing the incoming data from the barcode.

10.17.2.3 - (void) barcodeDataSent: (NSData *) *data* [optional]

This delegate monitors all data sent from Barcode Application.

Parameters

data	A NSData binary object representing the outgoing data sent the barcode.

10.17.2.4 - (void) barcodeInitialized: (BOOL) *isInitialized* [optional]

Notifies of initialization state changes in e210/e255/e315 Barcode Application.

Parameters

isInitialized	Device initialization state change • `TRUE` successfully initialized the barcode application, • `FALSE` barcode application went offline.

10.17.2.5 - (void) barcodeLogEntry: (NSString *) *logEntry* withSeverity:(int) *severity* [optional]

When logEnabled: (VFIBarcode) is passed `TRUE`, this delegate will receive log entries.

Parameters

logEntry	The log entry
severity	The severitry of the log entry, with 0 indicating highest priority

10.17.2.6 - (void) barcodeMessage: (NSString *) *message* [optional]

Additional messaging used by VMF.

Parameters

message	The message passed by VFIBarcode

10.17.2.7 - (void) barcodeReconnectFinished [optional]

This signifies the end of the framework initialization process for barcode application.

10.17.2.8 - (void) barcodeReconnectStarted [optional]

When the External Accessory reports the barcode devices is detected (but not initialized by framework),this delegate is called. This signifies the beginning of the framework initialization process of the barcode application.

10.17.2.9 - (void) barcodeScanData: (VFIEncryptedBarcode *) *data* [optional]

This delegate receives the encrypted barcode scan data.

Parameters

data	A VFIEncryptedBarcode class object representing the barcode data.

10.17.2.10 - (void) barcodeScanData: (NSData *) *data* barcodeType:(int) *thetype* [optional]

This delegate receives the barcode scan data.

If barcodeTypeEnabled: (VFIBarcode) is passed `TRUE`, then the parameter `thetype` will represent the barcode type int barcodeTypes.

Parameters

data	A NSData binary object representing the barcode data. • To convert the barcode data to a NSString, execute `[[NSString alloc] initWithData:data encoding:NSASCIIStringEncoding]`
thetype	An int value representing the barcode type captured, using enum barcodeTypes

10.17.2.11 - (void) barcodeScanData: (NSData *) *data* code:(int) *code* symbol:(int) *symbol* aim:(int) *aim* [optional]

This delegate receives the barcode scan data.

If barcodeTypeEnabled: (VFIBarcode) is passed `TRUE`, then the parameter `thetype` will represent the barcode type int barcodeTypes.

Parameters

data	A NSData binary object representing the barcode data. • To convert the barcode data to a NSString, execute `[[NSString alloc] initWithData:data encoding:NSASCIIStringEncoding]`
code	The Code ID value representing the barcode type captured
symbol	The Symbology ID value representing the barcode type captured
aim	The AIM ID value representing the barcode type captured

10.17.2.12 - (void) barcodeSerialData: (NSData *) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the Barcode application can be monitored through this delegate.

Parameters

data	The serial data represented as a NSData object
isIncoming	The direction of the data • `TRUE` specifies data being received from barcode application, • `FALSE` indicates data being sent to barcode application.

10.17.2.13 - (void) barcodeTriggerEvent: (int) *BCS_TRIGGER* [optional]

Receives int value triggerResponse.

When scanner is in Soft mode by executing setSoft (VFIBarcode), this delegate will receive trigger press/release events values triggerResponse.

Parameters

BCS_TRIGGER	triggerResponse

10.17.2.14 - (void) commandResult: (int) *result* [optional]

Receives int value commResult.

Monitoring this delegate will inform the success/failure from the Verix BCS application to API commands execute.

Parameters

result	commResult

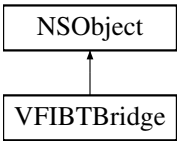
The documentation for this protocol was generated from the following file:

- VFIBarcode.h

10.18 VFIBTBridge Class Reference

```
#import <VFIBTBridge.h>
```

Inheritance diagram for VFIBTBridge:



Class Methods

- (NSString ∗) + frameworkVersion

  Framework Version.
- (void) + disableBTBridge

  Disable Bluetooth Bridge.
- (BOOL) + disabledBTBridge

  Disable Bluetooth Bridge Status.
- (BOOL) + notConnected
- (void) + setNoDisconnect:

  Control how framework handles connection when app goes to background.
- (BOOL) + noDisconnect

10.18.1 Method Documentation

10.18.1.1 + (void) disableBTBridge

Disable Bluetooth Bridge.

This method removes bluetooth support from VMF.

10.18.1.2 + (BOOL) disabledBTBridge

Disable Bluetooth Bridge Status.

This method returns bluetooth disabled flag.

10.18.1.3 + (NSString∗) frameworkVersion

Framework Version.

Returns VeriFone Mobile Framework Version.

10.18.1.4 + (BOOL) noDisconnect

10.18.1.5 + (BOOL) notConnected

10.18.1.6 + (void) setNoDisconnect: (BOOL) *val*

Control how framework handles connection when app goes to background.

When val is set to TRUE, all devices will stay connected in the background. Note: This method works for sled devices as well as Bluetooth.

The documentation for this class was generated from the following file:

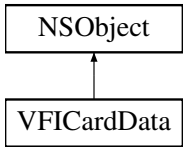
- VFIBTBridge.h

10.19 VFICardData Class Reference

Encapsulating data class utilized by VFIPinpad.

```
#import <VFICardData.h>
```

Inheritance diagram for VFICardData:



Instance Methods

- (void) - clear

  clears all VFICardData properties

Class Methods

- (VFICardData ∗) + sharedController

  Singleton instance of VFICardData utilized by VFIPinpad.

Properties

- NSString ∗ AID

  Application ID.
- NSString ∗ appPreferredName

  Application Preferred Name.
- NSString ∗ appLabel

  Application Label.
- int serviceCode

  Service Code.
- int entryType

  Pinpad entry type (swipe, keyed, clts)
- NSString ∗ cardHolderName

  Card Holder Name.
- NSData ∗ track2

  Track 2 data OR Track2 + Track 1 from S16 command.
- NSData ∗ track1

Track 1 data.

- NSData * track3

  Track 3 data.
- NSData * s16track1

  Track 1 data from S16 (obtainCardData)
- NSData * s16track2

  Track 2 data from S16 (obtainCardData)
- NSString * accountNumber

  Personal Account Number.
- NSString * expiryDate

  Expiration Date.
- BOOL isCTLS

  CTLS card used.
- NSMutableDictionary * emvTags

  Captured EMV Tags.

10.19.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.19.2 Method Documentation

10.19.2.1 - (void) clear

clears all VFICardData properties

10.19.2.2 + (VFICardData *) sharedController

Singleton instance of VFICardData utilized by VFIPinpad.

10.19.3 Property Documentation

10.19.3.1 - (NSString *) accountNumber [read], [write], [nonatomic], [retain]

Personal Account Number.

10.19.3.2 - (NSString *) AID [read], [write], [nonatomic], [retain]

Application ID.

10.19.3.3 - (NSString *) appLabel [read], [write], [nonatomic], [retain]

Application Label.

10.19.3.4 - (NSString *) appPreferredName [read], [write], [nonatomic], [retain]

Application Preferred Name.

10.19.3.5 - (NSString *) cardHolderName [read], [write], [nonatomic], [retain]

Card Holder Name.

10.19.3.6 - (NSMutableDictionary *) emvTags [read], [write], [nonatomic], [retain]

Captured EMV Tags.

10.19.3.7 - (int) entryType [read], [write], [atomic]

Pinpad entry type (swipe, keyed, clts)

10.19.3.8 - (NSString *) expiryDate [read], [write], [nonatomic], [retain]

Expiration Date.

10.19.3.9 - (BOOL) isCTLS [read], [write], [atomic]

CTLS card used.

10.19.3.10 - (NSData *) s16track1 [read], [write], [nonatomic], [retain]

Track 1 data from S16 (obtainCardData)

10.19.3.11 - (NSData *) s16track2 [read], [write], [nonatomic], [retain]

Track 2 data from S16 (obtainCardData)

10.19.3.12 - (int) serviceCode [read], [write], [atomic]

Service Code.

10.19.3.13 - (NSData *) track1 [read], [write], [nonatomic], [retain]

Track 1 data.

10.19.3.14 - (NSData *) track2 [read], [write], [nonatomic], [retain]

Track 2 data OR Track2 + Track 1 from S16 command.

10.19.3.15 - (NSData *) track3 [read], [write], [nonatomic], [retain]

Track 3 data.

The documentation for this class was generated from the following file:

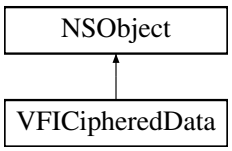
- VFICardData.h

10.20 VFICipheredData Class Reference

Encapsulating data class utilized by VFIPinpad.

```
#import <VFICipheredData.h>
```

Inheritance diagram for VFICipheredData:



Instance Methods

- (void) - clear

  clears all VFICipheredData properties

Class Methods

- (VFICipheredData ∗) + sharedController

  Singleton instance of VFICipheredData utilized by VFIPinpad.

Properties

- int encryptionType

  00 - None, 01 - VSP, 02 - PKI, 04- VSD DUKPT
- NSString ∗ keyID

  Key ID (variable, max up to 32 digits)
- int dataType

  Encryption mode.
- NSString ∗ encryptedBlob_1

  Ciphered Data, Track 2 Data, Track 1 Data.
- NSString ∗ encryptedBlob_2

  Ciphered Data (Optional, if Track3 is present) Track 3 Data.
- NSString ∗ VSD_Blob

  Ciphered Data, Base64 variable length encrypted blob containing VSD TLV list (VSD DUKPT encryption mode only)
- NSString ∗ EParms

  String of data returned by VCL that is then used by the VSD to determine information about the encryption status of the PAN data.
- NSString ∗ TKG1_Track1

  Terminal Generated Key 1, Track 1.
- NSString ∗ TKG1_Track2

  Terminal Generated Key 1, Track 2.
- NSString ∗ TKG2_Track1

  Terminal Generated Key 2, Track 1.
- NSString ∗ TKG2_Track2

Terminal Generated Key 2, Track 2.

- NSString * TKG3_Track1

  Terminal Generated Key 3, Track 1.

- NSString * TKG3_Track2

  Terminal Generated Key 3, Track 2.

10.20.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.20.2 Method Documentation

10.20.2.1 - (void) clear

clears all VFICipheredData properties

10.20.2.2 + (VFICipheredData *) sharedController

Singleton instance of VFICipheredData utilized by VFIPinpad.

10.20.3 Property Documentation

10.20.3.1 - (int) dataType [read], [write], [atomic]

Encryption mode.

10.20.3.2 - (NSString *) encryptedBlob_1 [read], [write], [nonatomic], [retain]

Ciphered Data, Track 2 Data, Track 1 Data.

10.20.3.3 - (NSString *) encryptedBlob_2 [read], [write], [nonatomic], [retain]

Ciphered Data (Optional, if Track3 is present) Track 3 Data.

10.20.3.4 - (int) encryptionType [read], [write], [atomic]

00 - None, 01 - VSP, 02 - PKI, 04- VSD DUKPT

10.20.3.5 - (NSString *) EParms [read], [write], [nonatomic], [retain]

String of data returned by VCL that is then used by the VSD to determine information about the encryption status of the PAN data.

10.20.3.6 - (NSString *) keyID [read], [write], [nonatomic], [retain]

Key ID (variable, max up to 32 digits)

10.20.3.7 - (NSString *) TKG1_Track1 [read], [write], [nonatomic], [retain]

Terminal Generated Key 1, Track 1.

10.20.3.8 - (NSString *) TKG1_Track2 [read], [write], [nonatomic], [retain]

Terminal Generated Key 1, Track 2.

10.20.3.9 - (NSString *) TKG2_Track1 [read], [write], [nonatomic], [retain]

Terminal Generated Key 2, Track 1.

10.20.3.10 - (NSString *) TKG2_Track2 [read], [write], [nonatomic], [retain]

Terminal Generated Key 2, Track 2.

10.20.3.11 - (NSString *) TKG3_Track1 [read], [write], [nonatomic], [retain]

Terminal Generated Key 3, Track 1.

10.20.3.12 - (NSString *) TKG3_Track2 [read], [write], [nonatomic], [retain]

Terminal Generated Key 3, Track 2.

10.20.3.13 - (NSString *) VSD_Blob [read], [write], [nonatomic], [retain]

Ciphered Data, Base64 variable length encrypted blob containing VSD TLV list (VSD DUKPT encryption mode only)

The documentation for this class was generated from the following file:

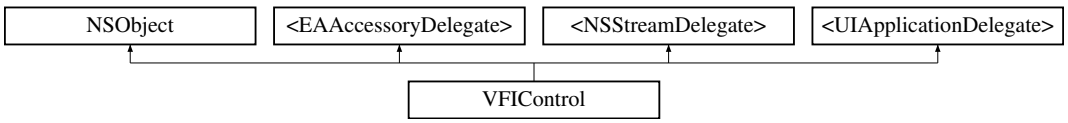
- VFICipheredData.h

10.21 VFIControl Class Reference

API methods for e210/e255/e315 Pinpad Control.

`#import <VFIControl.h>`

Inheritance diagram for VFIControl:



Instance Methods

- (id) - init

  Creates an instance of VFIControl class.
- (void) - initDevice

  Initializes the control application.
- (void) - initDeviceOnServer
- (void) - closeDevice

  Terminates stream connection to Control application.
- (void) - sendCommand:

  Send a string command to Control application.
- (void) - sendCommandLRC:

  Send a string command to Control application.
- (void) - disableKeypadOnUSBPower:

  API method to disable keypad when USB power is plugged in.
- (void) - keypadEnabled:

  API method to enable or disable the keypad.
- (void) - hostPowerEnabled:

  API method to enable or disable the iOS device charging from e210/e255/e315 battery.
- (void) - keypadBeepEnabled:

  API method to enable or disable the keypad beep that sounds whenever a key is pressed on the e210/e255/e315 PINPad.
- (void) - powerDown

  API method to power down e210/e255/e315.
- (void) - queryBatteryLevel

  Queries e210/e255/e315 for current battery level.
- (void) - querySoftwareVersion

  Queries e210/e255/e315 for current control software version.
- (void) - queryKeypadVersion

  Queries e210/e255/e315 for current e210/e255/e315 keypad version.
- (NSString *) - getKeypadState

  Get Keypad State.
- (NSString *) - getBTFirmwareVersion

  Get Bluetooth Firmware Version.
- (NSString *) - getBTFriendlyName

  Get Bluetooth Friendly Name.
- (NSString *) - getBTPIN

  Get Bluetooth PIN.
- (void) - rebootDevice

  Reboot Device.
- (void) - getGen3DeviceInfo

  Get Gen3 Device Info.
- (NSString *) - getKSIVersion

  Get KSI.
- (bool) - setDeviceLogging:

  Set Device Logging Status.
- (int) - queryDeviceLogging

  Get Device Logging Status.

- (NSString *) - getLogRecord

  Get Log Record.
- (NSString *) - getCrashInfo

  Get Crash Info.
- (bool) - enableLowBatteryNotification:

  Enable Low Battery Notification.
- (bool) - enableLowBatteryNotification:withNotificationLevel:

  Enable Low Battery Notification.
- (NSString *) - performMSRDiagnostic

  Perform MSR Diagnostics.
- (NSString *) - performPINEntryDiagnostic

  Perform PIN Entry Diagnostic.
- (bool) - performKeypadAllKeyDiagnostic

  Perform Keypad All Key Diagnostic.
- (bool) - performDisplayDiagnostic

  Perform Display Diagnostic.
- (bool) - startDiagnosticMode

  Start Diagnostic Mode.
- (void) - stopDiagnosticMode

  Stop Diagnostic Mode.
- (bool) - setDateAndTime:

  Set Date And Time.
- (NSString *) - getFileHash:

  Get File Hash.
- (bool) - setContactlessLED:

  Set Contactless LED.
- (BOOL) - isHostPowerEnabled_e335

  Host Device Charging in e335.
- (BOOL) - hostPowerEnabled_e335:

  Enable Host Device Charging in e335.
- (BOOL) - enableWatchdogTimer:withCountdownValue:

  iOS6 Watchdog Timer
- (BOOL) - displayMessage:withX:andY:shouldClearDisplay:

  Display Message.
- (BOOL) - queryDiagnosticMode

  Diagnostic Mode Query.
- (POWER_STATUS) - getPowerStatus

  Get Power Status.
- (BOOL) - getBTSPPEnable

  Get Bluetooth SPP Enable.
- (void) - consoleEnabled:

  Enabled logging to XCode Console.
- (void) - logEnabled:

  Enabled logging to delegate.
- (void) - restartLoopDelay:

  Controls the restart loop delay.
- (void) - setInitLoop:

Controls the amount of looping attempts to contact control application.

- (void) - enableBlocking

  Renables default blocking on all API methods.
- (void) - disableBlocking

  Disables default blocking on all API methods.
- (bool) - isHostPowerEnabled

  Get Power Mode.
- (void) - processReceivedData:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - ignoreDisconnect

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - disableProtocol:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - sendData:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Class Methods

- (VFIControl ∗) + sharedController

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Properties

- id< VFIControlDelegate > delegate

  Gets or Sets delegate for protocols.
- NSString ∗ controlName

  Read Only name reported to External Accessory.
- NSString ∗ controlManufacturer

  Read Only Manufacturer reported to External Accessory.
- NSString ∗ controlModelNumber

  Read Only Model Number reported to External Accessory.
- NSString ∗ controlSerialNumber

  Read Only Serial Number reported to ExternalExternal AccessoryAccessory.
- NSString ∗ controlFirmwareRevision

  Read Only Firmware Version reported to External Accessory.
- NSString ∗ controlHardwareRevision

  Read Only Hardware Version reported to External Accessory.
- int batteryLevel

  contains result after running queryBatteryLevel
- BOOL controlConnected

  Read Only Boolean control connection status.
- BOOL connected

  Read Only Boolean control connection status.
- EAAccessory ∗ eaACC

  Read Only Connected Accessory.
- BOOL BTconnected

Read Only Boolean barcode e210/e255/e315 Bluetooth connection Status.

- VFISoftwareVersion * vfiSoftwareVersion

  contains result after running querySoftwareVersion
- VFIKeypadVersion * vfiKeypadVersion

  contains result after running queryKeypadVersion
- BOOL initialized

  Read Only Boolean control initialized.
- BOOL isGen3

  Read Only Boolean Gen3 Connected.
- VFIGen3DeviceInfo * vfiGen3DeviceInfo

  Holds results from G25-getGen3DeviceInfo command.

10.21.1 Detailed Description

API methods for e210/e255/e315 Pinpad Control.

Implementing this class will allow access to API calls that will perform e210/e255/e315 Pinpad control, such as keypad beep on or off.

10.21.2 Method Documentation

10.21.2.1 - (void) closeDevice

Terminates stream connection to Control application.

This method will shut down the connection to the Control stream. An initDevice will need to be executed again to engage a new stream connection.

10.21.2.2 - (void) consoleEnabled: (BOOL) *enable*

Enabled logging to XCode Console.

Parameters

enable	Setting *TRUE* enables additional logging to debug console window in iOS

10.21.2.3 - (void) disableBlocking

Disables default blocking on all API methods.

By default blocking is on. Most API calls will wait for a response from Control app before returning control to integrator. This behavior can be turned off by calling this method. Default blocking can be turned back on by calling enableBlocking

10.21.2.4 - (void) disableKeypadOnUSBPower: (BOOL) *isDisabled*

API method to disable keypad when USB power is plugged in.

Parameters

isDisabled	• `TRUE` Disable keypad on USB Power • `FALSE` Enable keypad on USB Power

10.21.2.5 - (void) disableProtocol: (BOOL) *disable*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.21.2.6 - (BOOL) displayMessage: (NSString ∗) *message* withX:(NSString ∗) *x* andY:(NSString ∗) *y* shouldClearDisplay:(BOOL) *clearDisplay*

Display Message.

This command displays a message to the LCD display at the given location. This command is to be used in diagnostic mode only. Equivalent to G48 command.

Parameters

message	is a string to display at coordinates
x	is the x coordinate starting with column 1
y	is the y coordinate starting with column 1
clearDisplay	is a boolean value to indicate whether or not the screen should be cleared.

Return values

TRUE	= Command Success, FALSE = Command Failed

10.21.2.7 - (void) enableBlocking

Renables default blocking on all API methods.

By default blocking is on. Most API calls will wait for a response from Control app before returning control to integrator. This behavior can be turned off by calling disableBlocking

10.21.2.8 - (bool) enableLowBatteryNotification: (bool) *isEnabled*

Enable Low Battery Notification.

This command can enable and disable an asynchronous message from a Gen3 device to alert the application that the battery has reached a certain power threshold. This command will force the notification to only be returned at the 10% power level. The notification is asynchronous and will return in the controlBatteryEvent Delegate.

Note: You must be connected to the Control channel in order to receive this notification.

Parameters

isEnabled	• `TRUE` Enable low battery notification • `FALSE` Disable low battery notification

Return values

TRUE	= Enabled, FALSE = Error

10.21.2.9 - (bool) enableLowBatteryNotification: (bool) *isEnabled* withNotificationLevel:(int) *notificationLevel*

Enable Low Battery Notification.

This command can enable and disable an asynchronous message from a Gen3 device to alert the application that the battery has reached a certain power threshold. This command allows the integrator to provide a specific battery level for the low power event to fire at. Additionally, the notification will also fire at the 10% level as a specific low power indication. The notification is asynchronous and will return in the controlBatteryEvent Delegate.

Note: You must be connected to the Control channel in order to receive this notification.

Parameters

isEnabled	• `TRUE` Enable low battery notification • `FALSE` Disable low battery notification
notificationLevel	An integer to specify what battery level you would like the custom low battery event to fire at. Allowed values are 1-99.

Return values

TRUE	= Enabled, FALSE = Error

10.21.2.10 - (BOOL) enableWatchdogTimer: (BOOL) *disableTimer* withCountdownValue:(int) *countdownTimer*

iOS6 Watchdog Timer

This command is a work-around for an iOS6 bug in which the iAP2 connection cannot be re-established after an iOS6 App crashes. The default state is that the timer is disabled. When this command arrives, it sets the timer to the given number of milliseconds and starts the timer counting down. If the timer expires, the OS re-boots. Otherwise if another command arrives before the timer expires, the timer is reset to the new number of milliseconds. The timer can be shut off by sending this command with the “OFF” parameter. The valid timer settings are from 1 millisecond to 1 day. Equivalent to G51 command.

Parameters

disableTimer	is a boolean value to indicate if the workaround should be enabled.
countdownTimer	is a user provided int that indicates the length to wait.

Return values

TRUE	= Command Success, FALSE = Command Failed

10.21.2.11 - (NSString∗) getBTFirmwareVersion

Get Bluetooth Firmware Version.

Queries e255 to read the firmware version of the Bluetooth module.

Return values

An	ASCII string containing the version information.

10.21.2.12 - (NSString∗) getBTFriendlyName

Get Bluetooth Friendly Name.

This command allows the phone to read the friendly name of the Bluetooth module. The friendly name is the name that is displayed when discovering and pairing to e255. Use the ∗BT_NAME variable to change the name.

Return values

An	ASCII string containing the friendly name.

10.21.2.13 - (NSString∗) getBTPIN

Get Bluetooth PIN.

This command allows the phone to read the PIN number of the Bluetooth module. The pin number may not be required when Secure Simple Pairing is enabled. Use the ∗BT_PIN variable to change the PIN.

Return values

A	4 digit numeric ASCII PIN number string.

10.21.2.14 - (BOOL) getBTSPPEnable

Get Bluetooth SPP Enable.

This command allows the phone to read the enable state of the Secure Simple Pairing mode in the Bluetooth module. Enabling SSP allows some devices to pair to e255 without a PIN number. Some devices require SSP to be disabled in order to pair. Use the ∗BT_SSP_EN variable to change the setting.

Return values

TRUE	= SPP Enabled, FALSE = SPP Disabled

10.21.2.15 - (NSString∗) getCrashInfo

Get Crash Info.

This command retrieves system crash info from a Gen3 device.

Return values

A	string containing crash info

10.21.2.16 - (NSString∗) getFileHash: (NSString ∗) *filePath*

Get File Hash.

This command gets a SHA-256 hash of a given file on the Gen3 device.

Parameters

filePath	• filePath is an ASCII string containing the path for the Verix file.

Return values

NSString	with hash value or ERR if failed.

10.21.2.17 - (void) getGen3DeviceInfo

Get Gen3 Device Info.

This command returns a comma separated list of information fields for the Gen3 device. Note that some of the response properties will be deprecated in future updates.

Data returned from command will be stored in VFIGen3DeviceInfo object

10.21.2.18 - (NSString∗) getKeypadState

Get Keypad State.

Queries e210/e255/e315 for current keypad state. Result as returned as a two-digit NSString

Return values

2	Digit Code for Keypad State- Enable_State-Beep_State • Enable_State: 0 if keypad is disabled, 1 if keypad is enabled and awake, 2 if keypad is enabled, but still waking up. • Beep_State: 0 if beeps are disabled, 1 if beeps are enabled.

Example, "10" is keypad enabled, beeps disabled.

10.21.2.19 - (NSString∗) getKSIVersion

Get KSI.

This command returns the KSI fields of the keys in the 3 DUKPT engines of the IPP.

Return values

Three	comma separated ASCII hex fields. if no key is injected in that space, there will be no bytes returned. Commas will always be present.

10.21.2.20 - (NSString∗) getLogRecord

Get Log Record.

This command retrieves one log record at a time from a Gen3 device.

Return values

A	string containing a log record. If no record found, returns blank string.

10.21.2.21 - (POWER_STATUS) getPowerStatus

Get Power Status.

Returns IAP Manager power status.

Return values

A	POWER_STATUS structure populated with IAP Manager Power Status.

10.21.2.22 - (void) hostPowerEnabled: (BOOL) *isEnabled*

API method to enable or disable the iOS device charging from e210/e255/e315 battery.

Parameters

isEnabled	• `TRUE` iOS device will use e210/e255/e315 to charge internal iOS battery. • `FALSE` iOS device will NOT use e210/e255/e315 to charge internal iOS battery.

10.21.2.23 - (BOOL) hostPowerEnabled_e335: (BOOL) *isEnabled*

Enable Host Device Charging in e335.

This command will enable and disable host device charging. Host charging is enabled by setting configuration variable ∗ENHOSTCHARGE and disabled by clearing it. Restart required after enable and disable commands for OS to read change in configuration value. e335 Host Device Charging status can be queried via the isHostPowerEnabled_e335 command. Equivalent to G54 command.

Parameters

isEnabled	is a boolean value to indicate if host charging should be enabled on e335.

Return values

TRUE	= Command Success, FALSE = Command Failed

10.21.2.24 - (void) ignoreDisconnect

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.21.2.25 - (id) init

Creates an instance of VFIControl class.

Return values

<id>	of VFIControl class

Example Usage:

```
VFIControl* control = [[VFIControl alloc] init];
```

10.21.2.26 - (void) initDevice

Initializes the control application.

This is executed after the instance is created with init. If any of the optional protocols will be used, setDelegate should first be executed.

Example Usage:

```
VFIControl* control = [[VFIControl alloc] init];
[control setDelegate:self];
[control initDevice];
```

10.21.2.27 - (void) initDeviceOnServer

10.21.2.28 - (bool) isHostPowerEnabled

Get Power Mode.

This command returns the state of “Enable Host Power”.

Return values

Power	Mode: • FALSE: Power to host device disabled • TRUE: Host power enabled

10.21.2.29 - (BOOL) isHostPowerEnabled_e335

Host Device Charging in e335.

This command gives current status of host device charging for e335 ONLY. Host charging is enabled by setting configuration variable ∗ENHOSTCHARGE and disabled by clearing it and followed by enable call to hostPowerEnabled_e335. Equivalent to G54 command.

Return values

TRUE	= Host Charging is Enabled, FALSE = Host Charging is Disabled

10.21.2.30 - (void) keypadBeepEnabled: (BOOL) *isEnabled*

API method to enable or disable the keypad beep that sounds whenever a key is pressed on the e210/e255/e315 PINPad.

Parameters

isEnabled	• `TRUE` Enable beep • `FALSE` Disable beep

10.21.2.31 - (void) keypadEnabled: (BOOL) *isEnabled*

API method to enable or disable the keypad.

Parameters

isEnabled	• `TRUE` Enable keypad • `FALSE` Disable keypad

10.21.2.32 - (void) logEnabled: (BOOL) *enable*

Enabled logging to delegate.

Parameters

enable	Setting *TRUE* enables logging to controlLogEntry:withSeverity: (VFIControlDelegate-p)

10.21.2.33 - (bool) performDisplayDiagnostic

Perform Display Diagnostic.

This command tests display of Gen3 device by inverting all pixels for a few seconds.

Return values

TRUE	= Success, FALSE = Failed to open console device

10.21.2.34 - (bool) performKeypadAllKeyDiagnostic

Perform Keypad All Key Diagnostic.

This command prompts user to tap all keys on Gen3 keypad and display sucess on LCD screen when all key entries are detected. Otherwise displays failed message.

Return values

TRUE	= Success, FALSE = Failure

10.21.2.35 - (NSString∗) performMSRDiagnostic

Perform MSR Diagnostics.

This command returns status of MSR swipe and also each track status and number of bytes of track data. This command will timeout after 10 seconds.

Return values

Good	response is in ASCII hex format string. Bad response returns 'ERR' string. Timeout reached returns '-1' string.

10.21.2.36 - (NSString∗) performPINEntryDiagnostic

Perform PIN Entry Diagnostic.

This command prompts the user to tap 4 keys on Gen3 keypad and display success on LCD screen when 4 key entries are detected. Otherwise displays failed message.

Return values

A	12 byte string containing keypad firmware version

10.21.2.37 - (void) powerDown

API method to power down e210/e255/e315.

10.21.2.38 - (void) processReceivedData: (NSData ∗) *data*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.21.2.39 - (void) queryBatteryLevel

Queries e210/e255/e315 for current battery level.

Result in battery percentage 0-100 stored in VFIControl property batteryLevel()

10.21.2.40 - (int) queryDeviceLogging

Get Device Logging Status.

This command will return the logging status on the Gen3 device.

Return values

An	integer denoting the response. 1 = ENABLED, 0 = DISABLED, -1 = ERROR

10.21.2.41 - (BOOL) queryDiagnosticMode

Diagnostic Mode Query.

This command checks whether the Gen3 is in diagnostics mode. Equivalent to G47 command.

Return values

TRUE	= Device is in diagnostic mode, FALSE = Device is not in diagnostic mode

10.21.2.42 - (void) queryKeypadVersion

Queries e210/e255/e315 for current e210/e255/e315 keypad version.

Result as VFIKeypadVersion stored in VFIControl property vfiKeypadVersion()

10.21.2.43 - (void) querySoftwareVersion

Queries e210/e255/e315 for current control software version.

Result as VFISoftwareVersion stored in VFIControl property vfiSoftwareVersion()

10.21.2.44 - (void) rebootDevice

Reboot Device.

This command reboots the Verix OS.

10.21.2.45 - (void) restartLoopDelay: (float) *sec*

Controls the restart loop delay.

Parameters

sec	The amount of time in fractional sections to wait between attempts in establishing contact to e210/e255/e315 while waiting for initialization

Default is 1.0 seconds.

10.21.2.46 - (void) sendCommand: (NSString ∗) *cmd*

Send a string command to Control application.

Sends a command represented by the provide string object to the control application through the accessory protocol.

Parameters

cmd	NSString representation of command to execute

10.21.2.47 - (void) sendCommandLRC: (NSString ∗) *cmd*

Send a string command to Control application.

Sends a command represented by the provide command string to the control application through the accessory protocol. An LRC is calculated and appended to the command string sent.

Parameters

cmd	NSString representation of command to execute

10.21.2.48 - (void) sendData: (NSData ∗) *data*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.21.2.49 - (bool) setContactlessLED: (NSString ∗) *enableLEDString*

Set Contactless LED.

This command enables the LEDs of a gen3 device for use with contactless payments. Send a string of four bytes to determine which LEDs should be enabled or disabled. Send "0000" to turn off all LEDs and "1111" to turn on all LEDs.

Parameters

enableLED	is a string of four bytes that determines which LEDs should be on or off.

Return values

TRUE	= Success, FALSE = Failed to set LEDs

10.21.2.50 - (bool) setDateAndTime: (NSDate ∗) *date*

Set Date And Time.

This command sets date and time of Gen3. Date and time should be given in NSDate format and year must be in the range 1990 to 2089.

Parameters

date	• `NSDate*`

Return values

TRUE	= Set date successfully, FALSE = incorrect date/time was requested

10.21.2.51 - (bool) setDeviceLogging: (bool) *isEnabled*

Set Device Logging Status.

This command will enable and disable logging on the Gen3 device.

Parameters

isEnabled	• `TRUE` Enable device logging • `FALSE` Disable device logging

Return values

TRUE	= Command success, FALSE = Command failure

10.21.2.52 - (void) setInitLoop: (int) *loop*

Controls the amount of looping attempts to contact control application.

Parameters

loop	The number of loops in establishing contact to e210/e255/e315 while waiting for initialization

If the e210/e255/e315 is unresponsive, the framework will loop the specified number of times, with a delay of restartLoopDelay: between each attempt. The default is 59 loops.

10.21.2.53 + (VFIControl *) sharedController

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.21.2.54 - (bool) startDiagnosticMode

Start Diagnostic Mode.

This command displays diagnostic mode message on Gen3 LCD and sets the diagnostic flag.

Return values

TRUE	= Device is in Diagnostic Mode, FALSE = Error enabling diagnostic mode.

10.21.2.55 - (void) stopDiagnosticMode

Stop Diagnostic Mode.

This command exits diagnostic mode on the Gen3 and clears diagnostic mode flag. This command always returns OK response.

10.21.3 Property Documentation

10.21.3.1 - (int) batteryLevel [read], [atomic], [assign]

contains result after running queryBatteryLevel

10.21.3.2 - (BOOL) BTconnected [read], [atomic], [assign]

Read Only Boolean barcode e210/e255/e315 Bluetooth connection Status.

10.21.3.3 - (BOOL) connected [read], [atomic], [assign]

Read Only Boolean control connection status.

10.21.3.4 - (BOOL) controlConnected [read], [atomic], [assign]

Read Only Boolean control connection status.

10.21.3.5 - (NSString*) controlFirmwareRevision [read], [nonatomic], [assign]

Read Only Firmware Version reported to External Accessory.

10.21.3.6 - (NSString*) controlHardwareRevision [read], [nonatomic], [assign]

Read Only Hardware Version reported to External Accessory.

10.21.3.7 - (NSString*) controlManufacturer [read], [nonatomic], [assign]

Read Only Manufacturer reported to External Accessory.

10.21.3.8 - (NSString*) controlModelNumber [read], [nonatomic], [assign]

Read Only Model Number reported to External Accessory.

10.21.3.9 - (NSString*) controlName [read], [nonatomic], [assign]

Read Only name reported to External Accessory.

10.21.3.10 - (NSString*) controlSerialNumber [read], [nonatomic], [assign]

Read Only Serial Number reported to ExternalExternal AccessoryAccessory.

10.21.3.11 - (id< VFIControlDelegate >) delegate [read], [write], [atomic], [retain]

Gets or Sets delegate for protocols.

10.21.3.12 - (EAAccessory*) eaACC [read], [nonatomic], [assign]

Read Only Connected Accessory.

10.21.3.13 - (BOOL) initialized [read], [atomic], [assign]

Read Only Boolean control initialized.

10.21.3.14 - (BOOL) isGen3 [read], [atomic], [assign]

Read Only Boolean Gen3 Connected.

10.21.3.15 - (VFIGen3DeviceInfo*) vfiGen3DeviceInfo [read], [write], [nonatomic], [retain]

Holds results from G25-getGen3DeviceInfo command.

10.21.3.16 - (VFIKeypadVersion*) vfiKeypadVersion [read], [write], [nonatomic], [retain]

contains result after running queryKeypadVersion

10.21.3.17 - (VFISoftwareVersion*) vfiSoftwareVersion [read], [write], [nonatomic], [retain]

contains result after running querySoftwareVersion

The documentation for this class was generated from the following file:

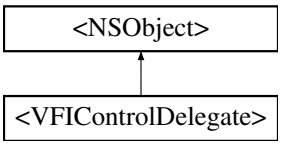
- VFIControl.h

10.22 <VFIControlDelegate> Protocol Reference

Protocol methods established for VFIControl class.

`#import <VFIControl.h>`

Inheritance diagram for <VFIControlDelegate>:



Instance Methods

- (void) - controlLogEntry:withSeverity:

  When logEnabled: (VFIControl) is passed `TRUE`, this delegate will receive log entries.
- (void) - controlInitialized:

  Notifies of initialization state changes in e210/e255/e315 Control Application.

- (void) - controlReconnectStarted

  When the External Accessory reports the control application is detected (but not initialized by framework),this delegate is called. This signifies the beginning of the framework initialization proess of the control application.
- (void) - controlReconnectFinished

  This signifies the end of the framework initialization process for control application.
- (void) - controlConnected:

  Notified of control connection/disconnection events.
- (void) - controlSerialData:incoming:

  All incoming/outgoing data going to the control application can be monitored through this delegate.
- (void) - controlDataReceived:

  This delegate monitors all data received from Control Application.
- (void) - controlDataSent:

  This delegate monitors all data sent from Control Application.
- (void) - controlBatteryLevel:
- (void) - controlSoftwareVersion:
- (void) - controlKeypadVersion:
- (void) - controlBatteryEvent:batteryEvent:

  Returns battery notification events if enabled.

10.22.1 Detailed Description

Protocol methods established for VFIControl class.

10.22.2 Method Documentation

10.22.2.1 - (void) controlBatteryEvent: (int) *batteryLevel* batteryEvent:(bool) *isLowBatteryEvent* [optional]

Returns battery notification events if enabled.

See Also

enableLowBatteryNotification() to enable these events. Events fire after the OS checks the battery level, which is every 10% or when external power is attached/removed. At 10% power remaining, a special low battery notification is sent, indicated by the isLowBatteryEvent flag.

Parameters

batteryLevel	Integer value of battery percentage between 0 and 100.
isLowBattery-Event	Indicates the type of event • `TRUE` specific low power event that fires at 10% remaining power • `FALSE` indicates that this is regular power event, either user specified level or device is unplugged/plugged in

10.22.2.2 - (void) controlBatteryLevel: (int) *batteryLevel* [optional]

Parameters

batteryLevel	A integer representing the battery level after calling

See Also

queryBatteryLevel().

10.22.2.3 - (void) controlConnected: (BOOL) *isConnected* [optional]

Notified of control connection/disconnection events.

A connect/disconnect can either be from a physical disconnection/connection with the External Accessory API, or from an application going to backround or returning to foreground.

Parameters

isConnected	A new connection or disconnection was detected • `TRUE` The control application has connected. • `FALSE` The control application has disconnected.

10.22.2.4 - (void) controlDataReceived: (NSData *) *data* [optional]

This delegate monitors all data received from Control Application.

Parameters

data	A NSData binary object representing the incoming data from the control application.

10.22.2.5 - (void) controlDataSent: (NSData *) *data* [optional]

This delegate monitors all data sent from Control Application.

Parameters

data	A NSData binary object representing the outgoing data sent the control application.

10.22.2.6 - (void) controlInitialized: (BOOL) *isInitialized* [optional]

Notifies of initialization state changes in e210/e255/e315 Control Application.

Parameters

isInitialized	Device initialization state change • `TRUE` successfully initialized the control application, • `FALSE` control application went offline.

10.22.2.7 - (void) controlKeypadVersion: (VFIKeypadVersion *) *keypadVersion* [optional]

Parameters

keypadVersion	VFIKeypadVersion() representing the keypad version after calling queryKeypadVersion().

10.22.2.8 - (void) controlLogEntry: (NSString *) *logEntry* withSeverity:(int) *severity* [optional]

When logEnabled: (VFIControl) is passed `TRUE`, this delegate will receive log entries.

Parameters

logEntry	The log entry
severity	The severitry of the log entry, with 0 indicating highest priority

10.22.2.9 - (void) controlReconnectFinished [optional]

This signifies the end of the framework initialization process for control application.

10.22.2.10 - (void) controlReconnectStarted [optional]

When the External Accessory reports the control application is detected (but not initialized by framework),this delegate is called. This signifies the beginning of the framework initialization proess of the control application.

10.22.2.11 - (void) controlSerialData: (NSData *) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the control application can be monitored through this delegate.

Parameters

data	The serial data represented as a NSData object
isIncoming	The direction of the data • `TRUE` specifies data being received from control application, • `FALSE` indicates data being sent to control application.

10.22.2.12 - (void) controlSoftwareVersion: (VFISoftwareVersion *) *softwareVersion* [optional]

Parameters

softwareVersion	VFISoftwareVersion() representing the software version after calling querySoftwareVersion().

The documentation for this protocol was generated from the following file:

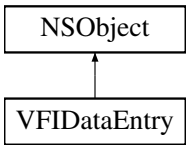
- VFIControl.h

10.23 VFIDataEntry Class Reference

Encapsulating data class utilized by VFIPinpad.

```
#import <VFIDataEntry.h>
```

Inheritance diagram for VFIDataEntry:



Instance Methods

- (void) - clear

  clears all VFIDataEntry properties

Class Methods

- (VFIDataEntry ∗) + sharedController

  Singleton instance of VFIDataEntry utilized by VFIPinpad.

Properties

- float tipAmount

  Tip amount entered (it can be in range 001 - 99999). 0 if not entered or bypassed. Implied decimal (example: 595 = 5.95).
- float cashbackAmount

  Cash Back amount (it can be in range 001 - 99999). 0 if not entered or bypassed. Implied decimal (example: 595 = 5.95).
- int surchargeIndicator

  1=Surcharge Accepted or Not Prompted, 2=Surcharge Denied
- NSString ∗ serialNumber

  9 digit serial number
- int amountConfirmed

  0= Amount Not Prompted, 1=Amount Prompted, 2=Amount Denied
- int accountSelected

  Account selected. 1=Checking, 2=Savings.
- NSData ∗ pinBlock

  16-character PIN Block (when applicable)
- int cashBackSelected

  0= CashBack Not Prompted, 1= CashBack prompted, 2=CashBack Denied
- int tipSelected

  0= Tip Not Prompted, 1=Tip prompted, 2=Tip Denied

10.23.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.23.2 Method Documentation

10.23.2.1 - (void) clear

clears all VFIDataEntry properties

10.23.2.2 + (VFIDataEntry *) sharedController

Singleton instance of VFIDataEntry utilized by VFIPinpad.

10.23.3 Property Documentation

10.23.3.1 - (int) accountSelected [read], [write], [atomic]

Account selected. 1=Checking, 2=Savings.

10.23.3.2 - (int) amountConfirmed [read], [write], [atomic]

0= Amount Not Prompted, 1=Amount Prompted, 2=Amount Denied

10.23.3.3 - (float) cashbackAmount [read], [write], [atomic]

Cash Back amount (it can be in range 001 - 99999). 0 if not entered or bypassed. Implied decimal (example: 595 = 5.95).

10.23.3.4 - (int) cashBackSelected [read], [write], [atomic]

0= CashBack Not Prompted, 1= CashBack prompted, 2=CashBack Denied

10.23.3.5 - (NSData *) pinBlock [read], [write], [nonatomic], [retain]

16-character PIN Block (when applicable)

10.23.3.6 - (NSString *) serialNumber [read], [write], [nonatomic], [retain]

9 digit serial number

10.23.3.7 - (int) surchargeIndicator [read], [write], [atomic]

1=Surcharge Accepted or Not Prompted, 2=Surcharge Denied

10.23.3.8 - (float) tipAmount [read], [write], [atomic]

Tip amount entered (it can be in range 001 - 99999). 0 if not entered or bypassed. Implied decimal (example: 595 = 5.95).

10.23.3.9 - (int) tipSelected [read], [write], [atomic]

0= Tip Not Prompted, 1=Tip prompted, 2=Tip Denied

The documentation for this class was generated from the following file:

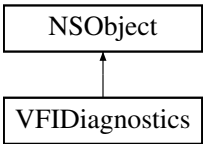
- VFIDataEntry.h

10.24 VFIDiagnostics Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIDiagnostics.h>`

Inheritance diagram for VFIDiagnostics:



Instance Methods

- (void) - clear

  clears all VFIDiagnostics properties

Class Methods

- (VFIDiagnostics ∗) + sharedController

  Singleton instance of VFIDiagnostics utilized by VFIPinpad.

Properties

- NSString ∗ osversion

  ROM OS Version.
- NSString ∗ xpiVersion

  Application Version.
- NSString ∗ vxciVersion

  Core Library Version.
- NSString ∗ emvVersion

  EMV Kernel version.
- NSString ∗ ctlsReaderFirmwareVersion

  CTLS Firmware Version.
- NSString ∗ pinpadSerialNumber

  Pinpad Serial Number.
- NSString ∗ svcInfoSerialNumber

  Pinpad Serial Number.

- NSString * svcInfoPartNumber

  Pinpad Serial Number.
- NSString * vspVersion

  VSP Firmware Version.

10.24.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.24.2 Method Documentation

10.24.2.1 - (void) clear

clears all VFIDiagnostics properties

10.24.2.2 + (VFIDiagnostics *) sharedController

Singleton instance of VFIDiagnostics utilized by VFIPinpad.

10.24.3 Property Documentation

10.24.3.1 - (NSString *) ctlsReaderFirmwareVersion [read], [write], [nonatomic], [retain]

CTLS Firmware Version.

10.24.3.2 - (NSString *) emvVersion [read], [write], [nonatomic], [retain]

EMV Kernel version.

10.24.3.3 - (NSString *) osversion [read], [write], [nonatomic], [retain]

ROM OS Version.

10.24.3.4 - (NSString *) pinpadSerialNumber [read], [write], [nonatomic], [retain]

Pinpad Serial Number.

10.24.3.5 - (NSString *) svcInfoPartNumber [read], [write], [nonatomic], [retain]

Pinpad Serial Number.

10.24.3.6 - (NSString *) svcInfoSerialNumber [read], [write], [nonatomic], [retain]

Pinpad Serial Number.

10.24.3.7 - (NSString *) vspVersion [read], [write], [nonatomic], [retain]

VSP Firmware Version.

This property will currently return null.

Reserved for future use.

10.24.3.8 - (NSString *) vxciVersion [read], [write], [nonatomic], [retain]

Core Library Version.

10.24.3.9 - (NSString *) xpiVersion [read], [write], [nonatomic], [retain]

Application Version.

The documentation for this class was generated from the following file:

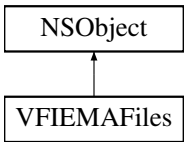
- VFIDiagnostics.h

10.25 VFIEMAFiles Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIEMAFiles.h>`

Inheritance diagram for VFIEMAFiles:



Instance Methods

- (void) - clear

  clears all VFIEMAFiles properties

Class Methods

- (VFIEMAFiles *) + sharedController

  Singleton instance of VFIEMAFiles utilized by VFIPinpad.

Properties

- int GID

  GID Number of EMA File (00-15)
- NSString * FileName

FileName of EMA File Retrieved including the Drive.

- NSString * Logical

  Logical Name.

- NSString * NAME

  Value of NAME in ApplicationData section from the EMA file.

- NSString * VERSION

  Value of VERSION in ApplicationData section from the EMA file.

- NSString * ALLOW_MESSAGES

  Value of ALLOW-MESSAGES in ApplicationData section from the EMA file.

- NSString * DOWNLOAD_ON_TIMEOUT

  Value of DOWNLOAD-ON-TIMEOUT in ApplicationData section from the EMA file.

10.25.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.25.2 Method Documentation

10.25.2.1 - (void) clear

clears all VFIEMAFiles properties

10.25.2.2 + (VFIEMAFiles *) sharedController

Singleton instance of VFIEMAFiles utilized by VFIPinpad.

10.25.3 Property Documentation

10.25.3.1 - (NSString *) ALLOW_MESSAGES [read], [write], [nonatomic], [retain]

Value of ALLOW-MESSAGES in ApplicationData section from the EMA file.

10.25.3.2 - (NSString *) DOWNLOAD_ON_TIMEOUT [read], [write], [nonatomic], [retain]

Value of DOWNLOAD-ON-TIMEOUT in ApplicationData section from the EMA file.

10.25.3.3 - (NSString *) FileName [read], [write], [nonatomic], [retain]

FileName of EMA File Retrieved including the Drive.

10.25.3.4 - (int) GID [read], [write], [atomic]

GID Number of EMA File (00-15)

10.25.3.5 - (NSString *) Logical [read], [write], [nonatomic], [retain]

Logical Name.

10.25.3.6 - (NSString *) NAME [read], [write], [nonatomic], [retain]

Value of NAME in ApplicationData section from the EMA file.

10.25.3.7 - (NSString *) VERSION [read], [write], [nonatomic], [retain]

Value of VERSION in ApplicationData section from the EMA file.

The documentation for this class was generated from the following file:

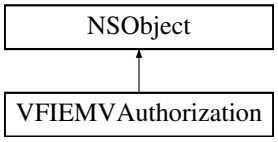
- VFIEMAFiles.h

10.26 VFIEMVAuthorization Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIEMVAuthorization.h>`

Inheritance diagram for VFIEMVAuthorization:



Instance Methods

- (void) - clear

  clears all VFIEMVAuthorization properties

Class Methods

- (VFIEMVAuthorization *) + sharedController

  Singleton instance of VFICardData utilized by VFIPinpad.

Properties

- NSMutableDictionary * emvTags

  A dictionary containing EMV Tag names and values, each of type NSData retrieved from the EMV chip card.
- int pinSuccessful

  Indicates the state of the PIN entry/validation for the card authorization; valid values are:

10.26.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.26.2 Method Documentation

10.26.2.1 - (void) clear

clears all VFIEMVAuthorization properties

10.26.2.2 + (VFIEMVAuthorization *) sharedController

Singleton instance of VFICardData utilized by VFIPinpad.

10.26.3 Property Documentation

10.26.3.1 - (NSMutableDictionary *) emvTags [read], [write], [nonatomic], [retain]

A dictionary containing EMV Tag names and values, each of type NSData retrieved from the EMV chip card.

10.26.3.2 - (int) pinSuccessful [read], [write], [atomic]

Indicates the state of the PIN entry/validation for the card authorization; valid values are:

- -1: N/A, PIN Successful field will only be returned if the Application Selection flag is not set to “S” or “E”. This field is required to indicate if Card Verification through PIN was successful or performed. The flag is required in Canada, to determine if a signature prints on the receipt, regardless of the Card’s decision if a signature is required or not.
- 0: PIN Entered Successfully
- 1: No PIN Entered
- 2: PIN Failed}

The documentation for this class was generated from the following file:

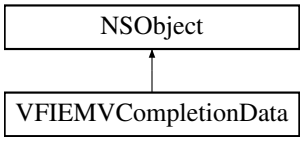
- VFIEMVAuthorization.h

10.27 VFIEMVCompletionData Class Reference

Encapsulating data class utilized by VFIPinpad.

```
#import <VFIEMVCompletionData.h>
```

Inheritance diagram for VFIEMVCompletionData:



Instance Methods

- (void) - clear

  clears all VFIEMVCompletionData properties

Class Methods

- (VFIEMVCompletionData ∗) + sharedController

  Singleton instance of VFIEMVCompletionData utilized by VFIPinpad.

Properties

- NSMutableDictionary ∗ emvTags

  A dictionary containing EMV Tag names and values, each of type NSData retrieved from the EMV chip card.
- int cardRemovalResult

  0 – Successful, card removed, 1 – Unsuccessful
- int writeToScriptResult

  Numeric response indicating the result of the writing of any script files to the inserted card.
- int validateMacResult

  Numeric response indicating the result of the request to the MAC validation request.

10.27.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.27.2 Method Documentation

10.27.2.1 - (void) clear

clears all VFIEMVCompletionData properties

10.27.2.2 + (VFIEMVCompletionData ∗) sharedController

Singleton instance of VFIEMVCompletionData utilized by VFIPinpad.

10.27.3 Property Documentation

10.27.3.1 - (int) cardRemovalResult [read], [write], [atomic]

0 – Successful, card removed, 1 – Unsuccessful

10.27.3.2 - (NSMutableDictionary *) emvTags [read], [write], [nonatomic], [retain]

A dictionary containing EMV Tag names and values, each of type NSData retrieved from the EMV chip card.

10.27.3.3 - (int) validateMacResult [read], [write], [atomic]

Numeric response indicating the result of the request to the MAC validation request.

- 00, Successful
- 01, Unsuccessful
- 05, Wrong Prefix (if balance is used, and failed) 07, MAC did not verify
- 09, No MAC block
- 11, Bad Key
- 14, Master key load failed.
- 16, New Working Key load failed. Revert successfully to the previous Working Keys set. 21, Cannot calculate MAC
- 23, Invalid field length
- 98, New Working Keys load failed. Cannot revert to the previous Working Keys

10.27.3.4 - (int) writeToScriptResult [read], [write], [atomic]

Numeric response indicating the result of the writing of any script files to the inserted card.

- 00, Successful
- 01, Unsuccessful
- 05, Wrong Prefix (if balance is used, and failed) 07, MAC did not verify
- 09, No MAC block
- 11, Bad Key
- 14, Master key load failed.
- 16, New Working Key load failed. Revert successfully to the previous Working Keys set. 21, Cannot calculate MAC
- 23, Invalid field length
- 98, New Working Keys load failed. Cannot revert to the previous Working Keys

The documentation for this class was generated from the following file:

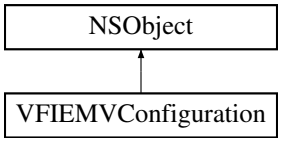
- VFIEMVCompletionData.h

10.28 VFIEMVConfiguration Class Reference

Encapsulating data class utilized by VFIPinpad.

```
#import <VFIEMVConfiguration.h>
```

Inheritance diagram for VFIEMVConfiguration:



Instance Methods

- (void) - clear

  clears all VFIEMVConfiguration properties

Properties

- int schemeReference

  2-byte int value
- int issuerReference

  2-byte int value
- int trmDataPresent

  2-byte int value
- int targetRSPercent

  2-byte int value
- double floorLimit

  4-byte int value
- double rsThreshold

  4-byte int value
- int maxTargetRSPercent

  2-byte int value
- int merchantForceOnlineFlag

  2-byte int value
- int blackListCardSupportFlag

  2-byte int value
- int fallbackAllowedFlag

  2-byte int value
- NSString * tacDefault

  ASCII string value.
- NSString * tacDenial

  ASCII string value.
- NSString * tacOnline

  ASCII string value.
- NSString * defaultTDOL

ASCII string value.

- NSString * defaultDDOL

  ASCII string value.
- int autoSelectApplication

  2-byte int value
- NSString * countryCode

  ASCII string value.
- NSString * currencyCode

  ASCII string value.
- NSString * terminalCapacity

  ASCII string value.
- NSString * additionalCapacity

  ASCII string value.
- NSString * terminalType

  ASCII string value.
- NSString * merchantCategoryCode

  ASCII string value.
- NSString * terminalCategoryCode

  ASCII string value.
- NSString * terminalID

  ASCII string value.
- NSString * merchantID

  ASCII string value.
- NSString * acquireID

  ASCII string value.
- NSString * capkIndex

  ASCII string value.
- NSString * pinBypassFlag

  ASCII string value.
- NSString * pinTimeout

  ASCII string value.
- NSString * pinFormat

  ASCII string value; '0'; 0-1 MS, 2-9 DUK, A EMV, B secure script.
- NSString * pinScriptNumber

  ASCII string value.
- NSString * pinMacroNumber

  ASCII string value.
- NSString * pinDevriKey

  ASCII string value; '0'; 0 - Traditional, 1 - Script, 2 - None.
- NSString * pinDevriMacro

  ASCII string value.
- NSString * cardStatusDisplay

  ASCII string value.
- int termCurExp

  2-byte int value
- int issAcqFlag

  2-byte int value

- int noDisplaySupportFlag

  2-byte int value
- int modifyCandidateListFlag

  2-byte int value
- NSString * rfu1String

  ASCII string value.
- NSString * rfu2String

  ASCII string value.
- NSString * rfu3String

  ASCII string value.
- int vfi_rfu1

  2-byte int value
- int vfi_rfu2

  2-byte int value
- int vfi_rfu3

  2-byte int value

10.28.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.28.2 Method Documentation

10.28.2.1 - (void) clear

clears all VFIEMVConfiguration properties

10.28.3 Property Documentation

10.28.3.1 - (NSString *) acquireID [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.2 - (NSString *) additionalCapacity [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.3 - (int) autoSelectApplication [read], [write], [atomic]

2-byte int value

10.28.3.4 - (int) blackListCardSupportFlag [read], [write], [atomic]

2-byte int value

10.28.3.5 - (NSString *) capkIndex [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.6 - (NSString *) cardStatusDisplay [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.7 - (NSString *) countryCode [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.8 - (NSString *) currencyCode [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.9 - (NSString *) defaultDDOL [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.10 - (NSString *) defaultTDOL [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.11 - (int) fallbackAllowedFlag [read], [write], [atomic]

2-byte int value

10.28.3.12 - (double) floorLimit [read], [write], [atomic]

4-byte int value

10.28.3.13 - (int) issAcqFlag [read], [write], [atomic]

2-byte int value

10.28.3.14 - (int) issuerReference [read], [write], [atomic]

2-byte int value

10.28.3.15 - (int) maxTargetRSPercent [read], [write], [atomic]

2-byte int value

10.28.3.16 - (NSString *) merchantCategoryCode [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.17 - (int) merchantForceOnlineFlag [read], [write], [atomic]

2-byte int value

10.28.3.18 - (NSString *) merchantID [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.19 - (int) modifyCandidateListFlag [read], [write], [atomic]

2-byte int value

10.28.3.20 - (int) noDisplaySupportFlag [read], [write], [atomic]

2-byte int value

10.28.3.21 - (NSString *) pinBypassFlag [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.22 - (NSString *) pinDevriKey [read], [write], [nonatomic], [retain]

ASCII string value; '0'; 0 - Traditional, 1 - Script, 2 - None.

10.28.3.23 - (NSString *) pinDevriMacro [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.24 - (NSString *) pinFormat [read], [write], [nonatomic], [retain]

ASCII string value; '0'; 0-1 MS, 2-9 DUK, A EMV, B secure script.

10.28.3.25 - (NSString *) pinMacroNumber [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.26 - (NSString *) pinScriptNumber [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.27 - (NSString *) pinTimeout [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.28 - (NSString *) rfu1String [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.29 - (NSString *) rfu2String [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.30 - (NSString *) rfu3String [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.31 - (double) rsThreshold [read], [write], [atomic]

4-byte int value

10.28.3.32 - (int) schemeReference [read], [write], [atomic]

2-byte int value

10.28.3.33 - (NSString *) tacDefault [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.34 - (NSString *) tacDenial [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.35 - (NSString *) tacOnline [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.36 - (int) targetRSPercent [read], [write], [atomic]

2-byte int value

10.28.3.37 - (int) termCurExp [read], [write], [atomic]

2-byte int value

10.28.3.38 - (NSString *) terminalCapacity [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.39 - (NSString *) terminalCategoryCode [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.40 - (NSString *) terminalID [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.41 - (NSString *) terminalType [read], [write], [nonatomic], [retain]

ASCII string value.

10.28.3.42 - (int) trmDataPresent [read], [write], [atomic]

2-byte int value

10.28.3.43 - (int) vfi_rfu1 [read], [write], [atomic]

2-byte int value

10.28.3.44 - (int) vfi_rfu2 [read], [write], [atomic]

2-byte int value

10.28.3.45 - (int) vfi_rfu3 [read], [write], [atomic]

2-byte int value

The documentation for this class was generated from the following file:

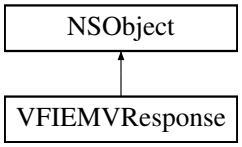
- VFIEMVConfiguration.h

10.29 VFIEMVResponse Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIEMVResponse.h>`

Inheritance diagram for VFIEMVResponse:



Instance Methods

- (void) - clear

  clears all VFIEMVResponse properties

Properties

- NSData ∗ macBlock

  8-character MAC block to verify
- NSData ∗ tpkKey

  PIN (TPK) Key.
- NSData ∗ takKey

  MAC (TAK) Key.
- NSData ∗ macData

  Buffer holding all data to be MAC'ed.
- int hostDecision

  Decision indicator from the host response; valid values are:
- BOOL displayResult

  Indicator to note whether or not to display the final response message;.
- NSData ∗ hostTagString

  A TLV (tag, length, value) data string received from the host response with EMV tags.
- int scriptId

  Numeric '71' or '72' value to be set based on the script sent from the host.
- BOOL clearBeforeScriptWrite

  Indicator to note whether or not to clear the script file before writing; a value of YES will clear before writing.
- int numberOfScripts

  The number of scripts included in the request.
- int scriptLength

  Length of the script data.
- NSData ∗ scriptData

  Hexadecimal script data of length scriptLength sent from the host for writing to the inserted card.

10.29.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.29.2 Method Documentation

10.29.2.1 - (void) clear

clears all VFIEMVResponse properties

10.29.3 Property Documentation

10.29.3.1 - (BOOL) clearBeforeScriptWrite [read], [write], [atomic]

Indicator to note whether or not to clear the script file before writing; a value of YES will clear before writing.

10.29.3.2 - (BOOL) displayResult [read], [write], [atomic]

Indicator to note whether or not to display the final response message;.

10.29.3.3 - (int) hostDecision [read], [write], [atomic]

Decision indicator from the host response; valid values are:

- 0:Host Authorized
- 1:Host Declined
- 2:Failed to Connect

10.29.3.4 - (NSData *) hostTagString [read], [write], [nonatomic], [retain]

A TLV (tag, length, value) data string received from the host response with EMV tags.

10.29.3.5 - (NSData *) macBlock [read], [write], [nonatomic], [retain]

8-character MAC block to verify

10.29.3.6 - (NSData *) macData [read], [write], [nonatomic], [retain]

Buffer holding all data to be MAC'ed.

10.29.3.7 - (int) numberOfScripts [read], [write], [atomic]

The number of scripts included in the request.

10.29.3.8 - (NSData *) scriptData [read], [write], [nonatomic], [retain]

Hexadecimal script data of length scriptLength sent from the host for writing to the inserted card.

10.29.3.9 - (int) scriptId [read], [write], [atomic]

Numeric '71' or '72' value to be set based on the script sent from the host.

10.29.3.10 - (int) scriptLength [read], [write], [atomic]

Length of the script data.

10.29.3.11 - (NSData *) takKey [read], [write], [nonatomic], [retain]

MAC (TAK) Key.

10.29.3.12 - (NSData *) tpkKey [read], [write], [nonatomic], [retain]

PIN (TPK) Key.

The documentation for this class was generated from the following file:

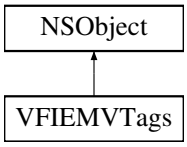
- VFIEMVResponse.h

10.30 VFIEMVTags Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIEMVTags.h>`

Inheritance diagram for VFIEMVTags:



Instance Methods

- (void) - clear

  clears all VFIEMVTags properties

Class Methods

- (VFIEMVTags *) + sharedController

  Singleton instance of VFIEMVTags utilized by VFIPinpad.

Properties

- NSMutableDictionary * emvTags

  A dictionary containing EMV Tag names and values, each of type NSData retrieved from the EMV chip card.

10.30.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.30.2 Method Documentation

10.30.2.1 - (void) clear

clears all VFIEMVTags properties

10.30.2.2 + (VFIEMVTags ∗) sharedController

Singleton instance of VFIEMVTags utilized by VFIPinpad.

10.30.3 Property Documentation

10.30.3.1 - (NSMutableDictionary ∗) emvTags `[read],[write],[nonatomic],[retain]`

A dictionary containing EMV Tag names and values, each of type NSData retrieved from the EMV chip card.

The documentation for this class was generated from the following file:

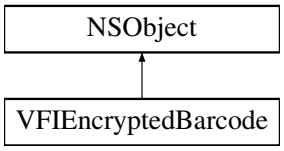
- VFIEMVTags.h

10.31 VFIEncryptedBarcode Class Reference

Encapsulating data class representing Encrypted Barcode Data.

`#import <VFIEncryptedBarcode.h>`

Inheritance diagram for VFIEncryptedBarcode:



Instance Methods

- (void) - clear

  clears all VFIEncryptedBarcode properties

Class Methods

- (VFIEncryptedBarcode ∗) + sharedController

  Singleton instance of VFIEncryptedBarcode utilized by VFIPinpad.

Properties

- NSString ∗ terminalSerialNumber

  Terminal Serial Number.

- NSString * publicKey

  Public Key ID.
- int scanIndex

  incremental barcode counter
- int decryptedLength

  Length of decrypted barcode data.
- NSMutableArray * encryptedData

  Array of NSData containg Base64 encoded encrypted binary data.

10.31.1 Detailed Description

Encapsulating data class representing Encrypted Barcode Data.

10.31.2 Method Documentation

10.31.2.1 - (void) clear

clears all VFIEncryptedBarcode properties

10.31.2.2 + (VFIEncryptedBarcode *) sharedController

Singleton instance of VFIEncryptedBarcode utilized by VFIPinpad.

10.31.3 Property Documentation

10.31.3.1 - (int) decryptedLength [read], [write], [atomic]

Length of decrypted barcode data.

10.31.3.2 - (NSMutableArray *) encryptedData [read], [write], [nonatomic], [retain]

Array of NSData containg Base64 encoded encrypted binary data.

10.31.3.3 - (NSString *) publicKey [read], [write], [nonatomic], [retain]

Public Key ID.

10.31.3.4 - (int) scanIndex [read], [write], [atomic]

incremental barcode counter

10.31.3.5 - (NSString *) terminalSerialNumber [read], [write], [nonatomic], [retain]

Terminal Serial Number.

The documentation for this class was generated from the following file:

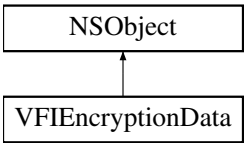
- VFIEncryptedBarcode.h

10.32 VFIEncryptionData Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIEncryptionData.h>`

Inheritance diagram for VFIEncryptionData:



Instance Methods

- (void) - clear

  clears all VFIEncryptionData properties

Class Methods

- (VFIEncryptionData ∗) + sharedController

  Singleton instance of VFIEncryptionData utilized by VFIPinpad.

Properties

- NSData ∗ macBlock

  8-character MAC Block
- NSData ∗ pinBlock

  16-character PIN Block as Data (when applicable)
- NSString ∗ pinBlockStr

  16-character PIN Block as String (when applicable)
- NSString ∗ serialNumber

  Serial Number of the VeriFone PINPad accessory.
- int accountSelection

  Account type selection, 1:checking, 2:savings.

10.32.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.32.2 Method Documentation

10.32.2.1 - (void) clear

clears all VFIEncryptionData properties

10.32.2.2 + (VFIEncryptionData *) sharedController

Singleton instance of VFIEncryptionData utilized by VFIPinpad.

10.32.3 Property Documentation

10.32.3.1 - (int) accountSelection [read], [write], [atomic]

Account type selection, 1:checking, 2:savings.

10.32.3.2 - (NSData *) macBlock [read], [write], [nonatomic], [retain]

8-character MAC Block

10.32.3.3 - (NSData *) pinBlock [read], [write], [nonatomic], [retain]

16-character PIN Block as Data (when applicable)

10.32.3.4 - (NSString *) pinBlockStr [read], [write], [nonatomic], [retain]

16-character PIN Block as String (when applicable)

10.32.3.5 - (NSString *) serialNumber [read], [write], [nonatomic], [retain]

Serial Number of the VeriFone PINPad accessory.

The documentation for this class was generated from the following file:

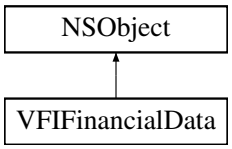
- VFIEncryptionData.h

10.33 VFIFinancialData Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIFinancialData.h>`

Inheritance diagram for VFIFinancialData:



Instance Methods

- (void) - clear

  clears all VFIFinancialData properties

Properties

- VFICardData * cardData

  Data collected from the swiped, inserted, or tapped card.
- VFIEncryptionData * encryptedData

  Encrypted data for the financial transaction, includes MAC and PIN blocks depending on the type of card used.
- NSMutableDictionary * emvTags

  EMV requested tags to be included in the financial transaction to the host.
- VFIEMVAuthorization * emvAuthorization

  EMV authorization results and additional requested tags for the financial transaction request to the host.
- VFIDataEntry * cardholderEntries

  Cardholder entered data in response to tip, cash back and surcharge prompting.
- NSMutableArray * additionalPrompts

  Additional cardholder prompts for which responses are required.
- NSMutableArray * additionalPromptResponses

  Cardholder value responses corresponding to the additionalPrompts.

10.33.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.33.2 Method Documentation

10.33.2.1 - (void) clear

clears all VFIFinancialData properties

10.33.3 Property Documentation

10.33.3.1 - (NSMutableArray *) additionalPromptResponses [read], [write], [nonatomic], [retain]

Cardholder value responses corresponding to the additionalPrompts.

10.33.3.2 - (NSMutableArray *) additionalPrompts [read], [write], [nonatomic], [retain]

Additional cardholder prompts for which responses are required.

10.33.3.3 - (VFICardData *) cardData [read], [write], [nonatomic], [retain]

Data collected from the swiped, inserted, or tapped card.

10.33.3.4 - (VFIDataEntry *) cardholderEntries [read], [write], [nonatomic], [retain]

Cardholder entered data in response to tip, cash back and surcharge prompting.

10.33.3.5 - (VFIEMVAuthorization *) emvAuthorization [read], [write], [nonatomic], [retain]

EMV authorization results and additional requested tags for the financial transaction request to the host.

10.33.3.6 - (NSMutableDictionary *) emvTags [read], [write], [nonatomic], [retain]

EMV requested tags to be included in the financial transaction to the host.

10.33.3.7 - (VFIEncryptionData *) encryptedData [read], [write], [nonatomic], [retain]

Encrypted data for the financial transaction, includes MAC and PIN blocks depending on the type of card used.

The documentation for this class was generated from the following file:

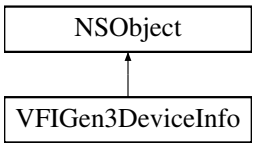
- VFIFinancialData.h

10.34 VFIGen3DeviceInfo Class Reference

Encapsulating data class utilized by VFIPinpad.

`#import <VFIGen3DeviceInfo.h>`

Inheritance diagram for VFIGen3DeviceInfo:



Instance Methods

- (void) - clear

  clears all VFIGen3DeviceInfo properties

Class Methods

- (VFIGen3DeviceInfo *) + sharedController

  Singleton instance of VFIGen3DeviceInfo utilized by VFIControl.

Properties

- NSString * bundleVersion

  Bundle Version ID.
- NSString * serialNumber

  Gen3 Serial Number.
- NSString * modelNumber

  Gen3 Model Number.

- NSString * partNumber

  Gen3 Part Number.
- NSString * hardwareVersion

  Gen3 Hardware Version.
- int batteryTemperature

  Current Battery Temperature.
- int batteryFullCharge

  Maximum Battery Charge Level.
- int batteryRemainingCharge

  Remaining Battery Charge Level.
- int batteryCurrent

  Charging Current Value (if charging)
- int batteryChargerStatus

  Returns Battery Charger Status Value.
- NSString * AMMVersion

  Returns AMM Version.
- NSString * SSLCLIVersion

  Returns SSLCLI Version.
- NSString * XPIVersion

  Returns XPI Version.

10.34.1 Detailed Description

Encapsulating data class utilized by VFIPinpad.

10.34.2 Method Documentation

10.34.2.1 - (void) clear

clears all VFIGen3DeviceInfo properties

10.34.2.2 + (VFIGen3DeviceInfo *) sharedController

Singleton instance of VFIGen3DeviceInfo utilized by VFIControl.

10.34.3 Property Documentation

10.34.3.1 - (NSString *) AMMVersion [read], [write], [nonatomic], [retain]

Returns AMM Version.

10.34.3.2 - (int) batteryChargerStatus [read], [write], [atomic]

Returns Battery Charger Status Value.

10.34.3.3 - (int) batteryCurrent [read], [write], [atomic]

Charging Current Value (if charging)

10.34.3.4 - (int) batteryFullCharge [read], [write], [atomic]

Maximum Battery Charge Level.

10.34.3.5 - (int) batteryRemainingCharge [read], [write], [atomic]

Remaining Battery Charge Level.

10.34.3.6 - (int) batteryTemperature [read], [write], [atomic]

Current Battery Temperature.

10.34.3.7 - (NSString *) bundleVersion [read], [write], [nonatomic], [retain]

Bundle Version ID.

10.34.3.8 - (NSString *) hardwareVersion [read], [write], [nonatomic], [retain]

Gen3 Hardware Version.

10.34.3.9 - (NSString *) modelNumber [read], [write], [nonatomic], [retain]

Gen3 Model Number.

10.34.3.10 - (NSString *) partNumber [read], [write], [nonatomic], [retain]

Gen3 Part Number.

10.34.3.11 - (NSString *) serialNumber [read], [write], [nonatomic], [retain]

Gen3 Serial Number.

10.34.3.12 - (NSString *) SSLCLIVersion [read], [write], [nonatomic], [retain]

Returns SSLCLI Version.

10.34.3.13 - (NSString *) XPIVersion [read], [write], [nonatomic], [retain]

Returns XPI Version.

The documentation for this class was generated from the following file:

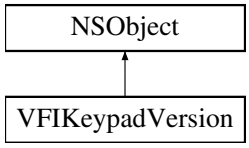
- VFIGen3DeviceInfo.h

10.35 VFIKeypadVersion Class Reference

Encapsulating data class utilized by VFIControl.

```
#import <VFIKeypadVersion.h>
```

Inheritance diagram for VFIKeypadVersion:



Instance Methods

- (void) - clear

  clears all VFIKeypadVersion properties

Properties

- NSString ∗ Processor

  Pinpad Processor.
- NSString ∗ BootloaderMajor

  Bootloader Version - Major.
- NSString ∗ BootloaderMinor

  Bootloader Version - Minor.
- NSString ∗ FirmwareMajor

  Keypad Firmware Version - Major.
- NSString ∗ FirmwareMinor

  Keypad Firmware Version - Minor.

10.35.1 Detailed Description

Encapsulating data class utilized by VFIControl.

10.35.2 Method Documentation

10.35.2.1 - (void) clear

clears all VFIKeypadVersion properties

10.35.3 Property Documentation

10.35.3.1 - (NSString ∗) BootloaderMajor [read], [write], [nonatomic], [retain]

Bootloader Version - Major.

10.35.3.2 - (NSString *) BootloaderMinor [read], [write], [nonatomic], [retain]

Bootloader Version - Minor.

10.35.3.3 - (NSString *) FirmwareMajor [read], [write], [nonatomic], [retain]

Keypad Firmware Version - Major.

10.35.3.4 - (NSString *) FirmwareMinor [read], [write], [nonatomic], [retain]

Keypad Firmware Version - Minor.

10.35.3.5 - (NSString *) Processor [read], [write], [nonatomic], [retain]

Pinpad Processor.

The documentation for this class was generated from the following file:

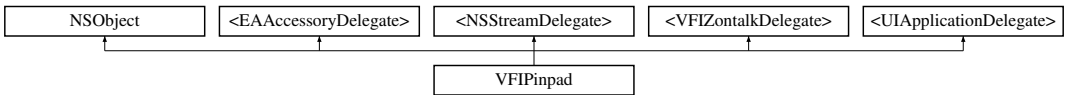
- VFIKeypadVersion.h

10.36 VFIPinpad Class Reference

API methods for e210/e255/e315 for XPI.

```
#import <VFIPinpad.h>
```

Inheritance diagram for VFIPinpad:



Instance Methods

- (id) - init

  Creates an instance of VFIPinpad class.
- (void) - setInitLoop:

  Controls the amount of looping attempts to contact XPI application.
- (void) - initServerOnPort:

  Initializes the XPI application server.
- (void) - initDevice

  Initializes the XPI application.
- (void) - initDeviceOnServer:port:
- (void) - closeDevice

  Terminates stream connection to XPI application.
- (void) - restartLoopDelay:

  Controls the restart loop delay.

- (void) - enableBlocking

  Renables default blocking on all API methods.
- (void) - disableBlocking

  Disables default blocking on all API methods.
- (void) - setKSN20Char:

  Forces Key Serial Number to 20 characters.
- (void) - updateFromUrl:

  Updates e210/e255/e315 from file specified by URL.
- (void) - updateFromZip:

  Updates e210/e255/e315 from a .zip file.
- (int) - sendStringCommand:calcLRC:

  Send a string command to XPI application.
- (int) - sendDataCommand:calcLRC:

  Send a NSData object to XPI application.
- (void) - setACKTimeout:

  Time to wait for `<ACK>` *in response to API calls sent to e210/e255/e315.*
- (void) - cancelPendingCommand

  Clears VMF buffer of any pending command not yet sent to e210/e255/e315 then executes cancelCommand
- (void) - cancelCommandWithDisplayOptions:

  Cancel Command With Display Options.
- (void) - cancelCommandWithDisplayOptions:andEMVFallback:disableKeyboard:

  Cancel Command With Display Options and EMV Fallback and Change Keyboard Status.
- (NSString *) - frameworkVersion

  Reports current framework version.
- (void) - setFrameworkTimeout:

  Time to wait for response from API calls sent to e210/e255/e315.
- (void) - setPromptTimeout:

  Time to wait for response to prompt.
- (void) - setPINTimeout:

  Time to wait for response to PIN entry.
- (void) - setAccountEntryTimeout:

  Time to wait for response to account number request.
- (void) - setContinueTransactionTimeout:

  Time to wait for response during EMV transaction.
- (void) - waitForResponse:

  Toggles API method calls blocking for next command.
- (void) - clearAllData

  Clears all data structures in VFIPinpad class.
- (void) - logEnabled:

  Enabled logging to delegate.
- (void) - consoleEnabled:

  Enabled logging to XCode Console.
- (void) - setParameterArray:

  Sends a list of parameters to the e210/e255/e315.
- (void) - setParameter:value:

  Sends a single parameters to the e210/e255/e315.
- (int) - D01:

Set PIN Bypass Support.

- (int) - pinBypass:

  Set PIN Bypass Support.
- (int) - S97

  Request EMA Files.
- (int) - requestEMAFiles

  Request EMA Files.
- (int) - S95Extended

  Polls e210/e255/e315 for Diagnostic Info.
- (int) - diagnosticInfoExtended

  Polls e210/e255/e315 for Diagnostic Info.
- (int) - S95

  Polls e210/e255/e315 for Diagnostic Info.
- (int) - diagnosticInfo

  Polls e210/e255/e315 for Diagnostic Info.
- (int) - clearCAPK

  Clear all CAPK Files.
- (int) - D12

  Clear all CAPK Files.
- (NSString ∗) - copyCAPKFile

  Filename of newly created CAPK from createCAPK()
- (NSString ∗) - copyStringResponse

  Input returned from requestStringInput()
- (NSDictionary ∗) - copyLastEMVTags

  Last EMV tags.
- (NSString ∗) - ctlsOPTFLAG

  OPTFLAG values from statusCTLS
- (int) - ctlsState

  Contactless reader state after running statusCTLS
- (void) - S01

  Query and Initialize Idle Prompt.
- (void) - initializeIdle

  Query and Initialize Idle Prompt.
- (int) - S68

  Test PIN Pad.
- (int) - testPINPad

  Test PIN Pad.
- (void) - Q41

  Toggle the Card Reader Off.
- (void) - disableMSR

  Toggle the Card Reader Off.
- (void) - Q40

  Toggle the Card Reader On.
- (void) - enableMSR

  Toggle the Card Reader On.
- (void) - Q42

  Toggle the Card Reader On.

- (void) - enableMSRDualTrack

  Toggle the Card Reader On.
- (void) - PS01:

  Set Currency Symbol.
- (void) - setCurrencySymbol:

  Set Currency Symbol.
- (int) - D11:pkIndex:modulus:exponent:checksum:expiryDate:

  Create CAPK File.
- (int) - D11a:pkIndex:modulus:exponent:checksum:expiryDate:

  Create CAPK File.
- (int) - createCAPK:pkIndex:modulus:exponent:checksum:expiryDate:

  Create CAPK File.
- (int) - createCAPKFile:pkIndex:modulus:exponent:checksum:expiryDate:

  Create CAPK File.
- (int) - D27

  Retrieve Status of Contactless Reader.
- (int) - statusCTLS

  Retrieve Status of Contactless Reader.
- (int) - D16:numRecords:

  Create EMV Tables.
- (int) - createTable:numRecords:

  Create EMV Tables.
- (int) - D18:sig:CVM:

  Initialize CTLS Reader.
- (int) - initializeContactless:sig:CVM:

  Initialize CTLS Reader.
- (int) - D26:aid:version:

  Update PassThru Configuration File.
- (int) - updatePassThru:aid:version:

  Update PassThru Configuration File.
- (int) - D25:ctlsMode:pdolSupport:valueLinkCtls:amexMode:

  Additional EMV Info.
- (int) - setAdditionalEVMFlags:ctlsMode:pdolSupport:valueLinkCtls:amexMode:

  Additional EMV Info.
- (int) - D25:ctlsMode:pdolSupport:

  Additional EMV Info.
- (int) - setAdditionalEVMFlags:ctlsMode:pdolSupport:

  Additional EMV Info.
- (int) - D13:AID:partialNameMatch:supportedAID:termAVN:term2ndAVN:recommendedAIDName:

  Update AID in EST (EMV Card Scheme Table)
- (int) - updateAIDRecord:AID:partialNameMatch:supportedAID:termAVN:term2ndAVN:recommendedAIDName:

  Update AID in EST (EMV Card Scheme Table)
- (int) - D14:tableData:

  Update AID in EST (EMV Card Scheme Table)
- (int) - updateEMVConfigRecord:tableData:

  Update MVT (EMV Configuration Table)

- (int) - D19:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:ctlsCountry:ctlsCurrency:

  Update EMVT.
- (int) - updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:ctlsCountry:ctlsCurrency:

  Update EMVT.
- (int) - D19:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:

  Update EMVT.
- (int) - updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:

  Update EMVT.
- (int) - D19:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:

  Update EMVT.
- (int) - updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:

  Update EMVT.
- (int) - D20:ctlsMerchantType:ctlsTermTransInfo:ctlsTermTransType:ctlsReqReceiptLimit:ctlsOptionStatus:ctlsReaderFlooorLimit:

  Update ICCT.
- (int) - updateICCT:ctlsMerchantType:ctlsTermTransInfo:ctlsTermTransType:ctlsReqReceiptLimit:ctlsOptionStatus:ctlsReaderFlooorLimit:

  Update ICCT.
- (int) - D17:emvRecord:AID:maxAIDLen:appFlow:partialName:acctSelect:enableCtls:transScheme:

  Update EEST Table.
- (int) - updateExtendedSchemeRecord:emvRecord:AID:maxAIDLen:appFlow:partialName:acctSelect:enableCtls:transScheme:

  Update EEST Table.
- (int) - D15:numberTrans:scheme:regId:CSNList:emvTableRecord:CSNFile:

  Update Scheme Record.
- (int) - updateSchemeRecord:numberTrans:scheme:regId:CSNList:emvTableRecord:CSNFile:

  Update Scheme Record.
- (EncryptionMode) - E00

  Get Encryption Mode.
- (EncryptionMode) - getEncryptionMode

  Get Encryption Mode.
- (int) - E02

  Retrieve EParms Data.
- (int) - retrieveEParmsData

  Retrieve EParms Data.
- (int) - E04

  Perform TGK Update Request.
- (int) - performTGKUpdateRequest

  Perform TGK Update Request.
- (int) - E06

  Ge Ciphered Data.
- (int) - getPKICipheredData

Get PKI Ciphered Data.

- (int) - getCipheredData

  Get Ciphered Data.
- (int) - E08_RSA:publicKeyID:

  Load Key Information.
- (int) - loadKeyInformation_RSA:publicKeyID:

  Get PKI Ciphered Data.
- (int) - E08_x509:

  Load Key Information.
- (int) - loadKeyInformation_x509:

  Get PKI Ciphered Data.
- (EncryptionMode) - E10:

  Select Encryption Mode.
- (EncryptionMode) - selectEncryptionMode:

  Select Encryption Mode.
- (EncryptionMode) - E10:keyIndex:

  Select Encryption Mode.
- (EncryptionMode) - selectEncryptionMode:keyIndex:

  Select Encryption Mode.
- (int) - E12:p75Data:

  Load PKI Signature Files.
- (int) - loadPKISig:p75Data:

  Load PKI Signature Files.
- (int) - E14

  Get Key ID Command.
- (int) - getKeyID

  Get Key ID Command.
- (void) - S00

  Cancel Command.
- (void) - cancelCommand

  Cancel Command.
- (int) - S11:line2:

  Display Message and Wait for Key.
- (int) - displayMessageAndWait:line2:

  Display Message and Wait for Key.
- (int) - S11:line2:line3:line4:

  Display Message and Wait for Key.
- (int) - displayMessageAndWait:line2:line3:line4:

  Display Message and Wait for Key.
- (void) - S07:pin:balance:

  Store Timer Values.
- (void) - storeTimers:pin:balance:

  Store Timer Values.
- (NSString *) - copyPromptData

  General Prompt Response.
- (int) - S12:message2:minLength:maxLength:language:allowSkip:secureEntry:displayCanceled:restoreScreen:

  General Prompt Entry.

- (int) - generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry:displayCanceled-:restoreScreen:

  General Prompt Entry.
- (int) - S12Default:message2:minLength:maxLength:language:initialValue:allowSkip:

  General Prompt Entry.
- (int) - generalPromptDefault:message2:minLength:maxLength:language:initialValue:allowSkip:

  General Prompt Entry.
- (int) - S12:message2:minLength:maxLength:language:allowSkip:secureEntry:

  General Prompt Entry.
- (int) - generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry:

  General Prompt Entry.
- (int) - S13:message2:choice1:choice2:choice3:choice4:

  Display Menu.
- (int) - menuChoiceSelection:message2:choice1:choice2:choice3:choice4:

  Display Menu.
- (void) - S14:Line2:Line3:Line4:timeout:

  Display Messages.
- (void) - displayMessages:Line2:Line3:Line4:timeout:

  Display Messages.
- (void) - S14:Line2:Line3:Line4:

  Display Messages.
- (void) - displayMessages:Line2:Line3:Line4:

  Display Messages.
- (int) - S16:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - obtainCardData:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - S16:displayIdle:displayExpiry:luhnCheck:entryMode:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - S16:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - S16:displayIdle:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - obtainCardData:displayIdle:

  Obtain Card Data (Manual Entry or Swipe)
- (int) - E20:

  Derived Key.
- (int) - derivedKey:

  Derived Key.
- (int) - S20:optionalAmount:

  Obtain Track 2 Data.
- (int) - obtainTrack2Data:optionalAmount:

  Obtain Track 2 Data.

- (int) - S21prompt:

  Data Entry Request.
- (int) - dataEntryRequestPrompt:

  Data Entry Request.
- (int) - S21:language:tip:cashback:surcharge:

  Data Entry Request.
- (int) - dataEntryRequest:language:tip:cashback:surcharge:

  Data Entry Request.
- (int) - S23:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBackOptions:surchargeOptions:

  Combined Prompt Command.
- (int) - combinedPromptCommand:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBackOptions:surchargeOptions:

  Combined Prompt Command.
- (int) - S66:

  Create Mac Request.
- (int) - createMac:

  Create Mac Request.
- (int) - S70:language:amount:pan:mac:

  Perform Interac Request Transaction.
- (int) - doInteracRequest:language:amount:pan:mac:

  Perform Interac Request Transaction.
- (int) - S71:language:pinKey:macKey:msgLine1:msgLine2:macData:

  Perform Interac Response Analysis.
- (int) - validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData:

  Perform Interac Response Analysis.
- (int) - S71Fields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK:

  Perform Interac Response Analysis.
- (int) - validateMacResponseFields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK:

  Perform Interac Response Analysis.
- (int) - C18:threshold:target:targetPercent:

  Modify TRM Parameters.
- (int) - setROSParams:threshold:target:targetPercent:

  Modify TRM Parameters.
- (int) - C19:value:

  Set EMV Tag Information.
- (int) - setEMVTag:value:

  Set EMV Tag Information.
- (int) - C20:

  Get EMV Tag Information.
- (int) - getEMVTag:

  Get EMV Tag Information.
- (int) - C21:

  Update EMV Tag Information.
- (int) - setEMVTags:

  Update EMV Tag Information.
- (int) - C25:clear:numScripts:length:data:

  Write to Script File.

- (int) - writeToScriptFile:clear:numScripts:length:data:

  Write to Script File.
- (int) - C30:language:amount:otherAmount:isRefundCTLS:disableCTLS:

  Enable EMV Entry.
- (int) - getCardData:language:amount:otherAmount:isRefundCTLS:disableCTLS:

  Get Card Data.
- (int) - setC30DefaultPrompts

  Prompt Message Changes.
- (int) - D30:line2:line3:line4:amt1:amt2:amt3:amt4:

  Prompt Message Changes.
- (int) - setC30CustomPrompts:line2:line3:line4:amt1:amt2:amt3:amt4:

  Prompt Message Changes.
- (int) - C30:language:amount:otherAmount:

  Enable EMV Entry.
- (int) - getCardData:language:amount:otherAmount:

  Write to Script File.
- (int) - C52:otherAmount:merchantDecision:minRequestObjects:

  PIN Management EMV Transaction Request.
- (int) - pinManagementCardAuthorization:otherAmount:merchantDecision:minRequestObjects:

  EMV Transaction Request.
- (int) - C32:otherAmount:merchantDecision:minRequestObjects:

  EMV Transaction Request.
- (int) - cardAuthorization:otherAmount:merchantDecision:minRequestObjects:

  EMV Transaction Request.
- (int) - C32:otherAmount:merchantDecision:minRequestObjects:displayResult:

  EMV Transaction Request.
- (int) - cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:

  EMV Transaction Request.
- (int) - C32:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:

  EMV Transaction Request.
- (int) - cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:

  EMV Transaction Request.
- (int) - C32:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:displayAppExpired:

  EMV Transaction Request.
- (int) - cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:displayAppExpired:

  EMV Transaction Request.
- (int) - C34:displayResult:hostTags:

  Complete Online EMV Transaction.
- (int) - completeOnlineEMV:displayResult:hostTags:

  Complete Online EMV Transaction.
- (int) - C54:displayResult:hostTags:

  Complete Online EMV Transaction for PIN Management.
- (int) - pinManagementCompleteOnlineEMV:displayResult:hostTags:

Complete Online EMV Transaction for PIN Management.

- (int) - C36:

  EMV Transaction Request.
- (int) - getEMVTags:

  EMV Transaction Request.
- (int) - I02

  Card Remove Request.
- (int) - removeCard

  Card Remove Request.
- (void) - Z2:clearScreen:

  Display a String.
- (void) - displayMessage:clearScreen:

  Display a String.
- (int) - Z56:message1:message2:

  Amount Entry.
- (int) - amountEntry:message1:message2:

  Amount Entry.
- (int) - Z58:amountFormat:message1:message2:

  Amount Introduction.
- (int) - amountIntroduction:amountFormat:message1:message2:

  Amount Introduction.
- (void) - Z3:message1:message2:clearScreen:

  Display Messages.
- (void) - displayMessages:message1:message2:clearScreen:

  Display Messages.
- (int) - Z50:timeout:maxLen:entryType:

  Request String Input.
- (int) - requestStringInput:timeout:maxLen:entryType:

  Request String Input.
- (int) - Z50:timeout:maxLen:

  Request String Input.
- (int) - requestStringInput:timeout:maxLen:

  Request String Input.
- (int) - Z52:echo:timeout:maxLen:entryType:

  Display Message and Prompt for Input.
- (int) - displayMessagePromptInput:echo:timeout:maxLen:entryType:

  Display Message and Prompt for Input.
- (void) - PS02:
- (void) - setIOTimer:
- (int) - Z60:

  Accept and Encrypt PIN.
- (int) - acceptEncryptPIN:

  Accept and Encrypt PIN.
- (int) - Z62:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processingMessage:

  Accept and Encrypt PIN, Display Custom Messages.
- (int) - acceptEncryptPINMessage:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processing-Message:

Accept and Encrypt PIN, Display Custom Messages.

- (int) - S67:

  Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.
- (int) - requestPINRentry:

  Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.
- (int) - S67:pinpadPrompt:timeout:

  Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.
- (int) - requestPINRentry:pinpadPrompt:timeout:

  Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.
- (void) - startServer

  Start the TCP server to listen for connections.
- (void) - stopServer

  Stops the TCP server from listening for connections. Terminates any current connection.
- (NSString ∗) - getOSVersion

  M01 returns OS version.
- (NSString ∗) - getXPIVersion

  M01 returns XPI Version.
- (void) - noResponseNextCommand
- (void) - processReceivedData:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - ignoreDisconnect

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - cancelPendingCommandNoData

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (void) - writeToSock:

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Class Methods

- (VFIPinpad ∗) + sharedController

  INTERNAL USE ONLY: Allows access for VFIBTBridge calls.
- (BOOL) + isGen3Server

Properties

- id< VFIPinpadDelegate > delegate

  Gets or Sets delegate for protocols.
- VFICardData ∗ vfiCardData

  Holds card data results from executing obtainCardData:, obtainTrack2Data:optionalAmount:, or getCardData:language:amount:otherAmount:
- VFIDataEntry ∗ vfiDataEntry

  Holds prompt results from executing S21prompt:, dataEntryRequest:language:tip:cashback:surcharge:, combinedPromptCommand:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBackOptions:surchargeOptions:
- VFICipheredData ∗ vfiCipheredData

  Holds cyphered results from executing retrieveEParmsData, performTGKUpdateRequest, getPKICipheredData, getCipheredData
- VFIEMVTags ∗ vfiEMVTags

Holds prompt results from executing getEMVTags:

- VFIDiagnostics * vfiDiagnostics

  Holds result from executing diagnosticInfo
- VFIEMVAuthorization * vfiEMVAuthorization

  Holds result from executing cardAuthorization:otherAmount:merchantDecision:minRequestObjects:
- VFIEMVCompletionData * vfiEMVCompletionData

  Holds result from executing completeOnlineEMV:displayResult:hostTags:
- VFIEncryptionData * vfiEncryptionData

  Holds result from executing testPINPad, createMac:, doInteracRequest:language:amount:pan:mac:, validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData:, acceptEncryptPIN:, acceptEncryptPINMessage:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processingMessage:
- NSString * pinpadName

  Gets or Sets delegate for protocols.
- NSString * pinpadManufacturer

  Read Only name reported to External Accessory.
- NSString * pinpadModelNumber

  Read Only Manufacturer reported to External Accessory.
- NSString * pinpadSerialNumber

  Read Only Serial Number reported to External Accessory.
- NSString * pinpadFirmwareRevision

  Read Only Firmware Version reported to External Accessory.
- NSString * pinpadHardwareRevision

  Read Only Hardware Version reported to External Accessory.
- NSArray * emaFiles

  Array of VFIEMAFiles containing the response from executing requestEMAFiles
- int ackTimeout

  Read Only Timeout Value waiting for pinPad to return an <ACK> from an executed command. Set with setResponseTimeout:()
- int frameworkTimeout

  Read Only Timeout Value waiting for pinPad to return a reply from an executed command. Set with setFrameworkTimeout:
- int ioTimer

  Read Only Timeout Value XPI awaiting input/output. Set with setIOTimer:
- int PINTimeout

  Read Only Timeout Value waiting for PIN entry during EMV transaction. Set with setPINTimeout:
- int vsdKeyIndex

  Read Only Index of DUKPT key when VSD encryption is enabled. Set after calling setPINTimeout:
- int accountEntryTimeout

  Read Only Timeout Value waiting for account entry during EMV transaction. Set with accountEntryTimeout:()
- int continueTransactionTimeout

  Read Only Timeout Value waiting to complete online EMV transaction. Set with continueTransactionTimeout:()
- int amountResponse

  Read Only value returned with successful response code to amountIntroduction:format:message1:message2:timeout:()
- NSData * emvTagData

  Read Only value returned with successful response code to amountIntroduction:format:message1:message2:timeout:()
- BOOL pinpadConnected

  Read Only Boolean pinpad connection status.
- BOOL connected

  Read Only Boolean pinpad connection status.

- EAAccessory * eaACC

  Read Only Connected Accessory.
- BOOL serverRunning

  Read Only Boolean TCP server status.
- BOOL BTconnected

  Read Only Boolean pinpad e210/e255/e315 Bluetooth connection Status.
- BOOL initialized

  Read Only Boolean pinpad initialized.
- AsyncSocket * listenSocket

  PRIVATE: Used by VMF to manage TCP Server Mode.
- AsyncSocket * currentSocket

  PRIVATE: Used by VMF to manage TCP Server Mode.
- NSMutableArray * connectedSockets

  PRIVATE: Used by VMF to manage TCP Server Mode.
- BOOL isRunning

  PRIVATE: Used by VMF to manage TCP Server Mode.
- BOOL remoteMode

  PRIVATE: Used by VMF to manage VMF Remote Mode.
- BOOL isVMFRemote

  PRIVATE: Used by VMF to manage VMF Remote Mode.
- BOOL isGen3

  Read Only Boolean Gen3 Connected.
- NSString * keyLoadErrorCode

  Error code returned from unsuccessful Key Loading.

10.36.1 Detailed Description

API methods for e210/e255/e315 for XPI.

Implementing this class will allow access to API calls that will perform e210/e255/e315 XPI commands.

10.36.2 Method Documentation

10.36.2.1 - (int) acceptEncryptPIN: (NSString *) *accountNumber*

Accept and Encrypt PIN.

This command is sent by the master device after initializing the PINPad to display one or two messages requesting a customer PIN number using Z2 displayMessage:clearScreen: or Z3 displayMessages:message1:message2:clearScreen:. Upon receipt of this packet and the customer PIN number, the PINPad will form a formatted clear-text PIN block based on the customer account number and PIN number. The PINPad encrypts this formatted clear-text PIN block to come up with the cipher-text PIN block returned.

The resulting data from the execution of this command can be retrieved from vfiEncryptionData, specifically, the serialNumber and pinBlock member variables

XPI reference: Z60.

Parameters

accountNumber	The account number used with the entered PIN to create the encrypted PIN Block (8 to 19 numeric digits)

Return values

Result	Code: • 00: Completed. Check for valid data (if any) in vfiEncryptionData

10.36.2.2 - (int) acceptEncryptPINMessage: (NSString ∗) *accountNumber* minPIN:(int) *min* maxPIN:(int) *max* requirePIN:(BOOL) *req* firstMessage:(NSString ∗) *msg1* secondMessage:(NSString ∗) *msg2* processingMessage:(NSString ∗) *procMsg*

Accept and Encrypt PIN, Display Custom Messages.

This command is sent by the calling application to accept a PIN number from a customer, build a PIN block, encrypt it, and return to the calling application. Upon receipt of this packet and the customer PIN number, the PINPad will form a formatted clear-text PIN block based on the customer account number and PIN number. The PINPad encrypts this formatted clear-text PIN block to come up with the cipher-text PIN block returned. This command also allows the calling application to specify the length of the PIN number and whether or not a NULL PIN entry is allowed. It displays up to 2 messages, alternating with each other, before PIN entry and a processing message displayed after PIN entry.

The resulting data from the execution of this command can be retrieved from vfiEncryptionData, specifically, the serialNumber and pinBlock member variables

XPI reference: Z62.

Parameters

accountNumber	The account number used with the entered PIN to create the encrypted PIN Block (8 to 19 numeric digits)
min	PIN Length
max	PIN Length
req	empty/no PIN
msg1	Message 1 to be displayed on the PINPad (alternates with Message 2)
msg2	Message 2 to be displayed on the PINPad (alternates with Message 1)
procMsg	Processing message to be displayed after PIN entry

Return values

Result	Code: • 00: Completed. Check for valid data (if any) in vfiEncryptionData

10.36.2.3 - (int) amountEntry: (int) *amount* message1:(NSString ∗) *msg1* message2:(NSString ∗) *msg2*

Amount Entry.

This command will display an amount, and will prompt for an amount. If there is a successful response code (00), then the informatted amounted entered can be found in amountResponse property.

Parameters

amount	Send whole number - two decimal places will be assumed (1295=12.95). This field is not required. The user can just display a message without an amount. Send 0 to not display amount.
msg1	Text can be passed into XPI. The message will not need to be verified against the secure message file since the entry is an amount entry. This field is not required.
msg2	Display the message for the amount prompt. This message is required.

- Secure Message Table

XPI reference: Z56

Return values

Result	Code: • 00: Successful. Unformatted amount entered is returned in property amountResponse • 01: Invalid Parameter • 02: Timed out • 03: CANCEL key was pressed • 04: User Bypass (applicable for amount equals to zero)

10.36.2.4 - (int) amountIntroduction: (int) *defaultValue* amountFormat:(int) *format* message1:(NSString ∗) *msg1* message2:(NSString ∗) *msg2*

Amount Introduction.

This command is used to prompt XPI for the Amount Introduction. The prompt messages allowed to be used on this command are only those messages that are included in the list of Secure Messages. If there is a successful response code (00), then the informatted amounted entered can be found in amountResponse property.

Parameters

defaultValue	Default Amount to be displayed (according to format)
format	0 or 1. Using a default value example of 123456789: • Format 0 : 1,234,567.89 • Format 1 : 1 234 567,89 • Note: The currency symbol to use is defined by the XPICURR config setting

msg1	First secure message to display
msg2	Second secure message to display

- Secure Message Table

XPI reference: Z58

Return values

Result	Code: • 00: Successful. Unformatted amount entered is returned in property amount-Response • 01: Invalid Parameter • 02: Timed out • 03: CANCEL key was pressed • 04: User Bypass (applicable for amount equals to zero)

10.36.2.5 - (int) C18: (uint) *floorLimit* threshold:(uint) *thresh* target:(uint) *targ* targetPercent:(uint) *percent*

Modify TRM Parameters.

See Also

- setROSParams:threshold:target:targetPercent:

10.36.2.6 - (int) C19: (NSString ∗) *tag* value:(NSData ∗) *tagValue*

Set EMV Tag Information.

See Also

- setEMVTag:value:

10.36.2.7 - (int) C20: (NSString ∗) *tag*

Get EMV Tag Information.

See Also

- getEMVTag:

10.36.2.8 - (int) C21: (NSMutableDictionary ∗) *tagArray*

Update EMV Tag Information.

See Also

- setEMVTags:

10.36.2.9 - (int) C25: (int) *scriptId* clear:(BOOL) *flag* numScripts:(int) *num* length:(int) *scriptLen* data:(NSData ∗) *scriptData*

Write to Script File.

See Also

- writeToScriptFile:clear:numScripts:length:data:

10.36.2.10 - (int) C30: (int) *timeout* language:(int) *lcode* amount:(float) *txnAmt* otherAmount:(float) *txnOthAmt*

Enable EMV Entry.

See Also

- getCardData:language:amount:otherAmount:

10.36.2.11 - (int) C30: (int) *timeout* language:(int) *lcode* amount:(float) *txnAmt* otherAmount:(float) *txnOthAmt* isRefundCTLS:(BOOL) *isRefund* disableCTLS:(BOOL) *disableCTLS*

Enable EMV Entry.

See Also

- getCardData:language:amount:otherAmount:isRefundCTLS:disableCTLS:

10.36.2.12 - (int) C32: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray*

EMV Transaction Request.

See Also

- cardAuthorization:otherAmount:merchantDecision:minRequestObjects:

10.36.2.13 - (int) C32: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray* displayResult:(BOOL) *display*

EMV Transaction Request.

See Also

- cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:

10.36.2.14 - (int) C32: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray* displayResult:(BOOL) *display* pinTryExceedStatus:(BOOL) *pinTry* displayAmount:(BOOL) *displayAmt*

EMV Transaction Request.

See Also

- cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:

10.36.2.15 - (int) C32: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray* displayResult:(BOOL) *display* pinTryExceedStatus:(BOOL) *pinTry* displayAmount:(BOOL) *displayAmt* displayAppExpired:(BOOL) *displayAppExpired*

EMV Transaction Request.

See Also

- cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:pinTryExceedStatus:displayAmount:displayAppExpired:

10.36.2.16 - (int) C34: (int) *hostDecision* displayResult:(BOOL) *flag* hostTags:(NSMutableDictionary ∗) *hostTagData*

Complete Online EMV Transaction.

See Also

- completeOnlineEMV:displayResult:hostTags:

10.36.2.17 - (int) C36: (NSMutableArray ∗) *tagArray*

EMV Transaction Request.

See Also

- getEMVTags:

10.36.2.18 - (int) C52: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray*

PIN Management EMV Transaction Request.

See Also

- pinManagementCardAuthorization:otherAmount:merchantDecision:minRequestObjects:

10.36.2.19 - (int) C54: (int) *hostDecision* displayResult:(BOOL) *flag* hostTags:(NSMutableDictionary ∗) *hostTagData*

Complete Online EMV Transaction for PIN Management.

See Also

- pinManagementCompleteOnlineEMV:displayResult:hostTags:

10.36.2.20 - (void) cancelCommand

Cancel Command.

This command will cancel any PIN pad command that is pending User Action such as:

- Card insertion or card swipe
- Data Entry or Function Key
- Signature Capture Request

It is recommended that the host device send an S01 command after the S00 command if the WELCOME/BONJOUR message is to appear. S00 Command cancels both US and Canadian commands. S00 will not display TRANSACTION CANCELED if sent during C30 or S20 commands.

XPI reference: S00

10.36.2.21 - (void) cancelCommandWithDisplayOptions: (int) *display*

Cancel Command With Display Options.

This command will cancel any PIN pad command that is pending User Action such as:

- Card insertion or card swipe
- Data Entry or Function Key
- Signature Capture Request

Parameters

display	int value to specify what the screen should display after this command • 0: No Screen Change • 1: Display Welcome Message • 2: Clear The Screen

It is recommended that the host device send an S01 command after the S00 command if the WELCOME/BONJOUR message is to appear. S00 Command cancels both US and Canadian commands. S00 will not display TRANSACTION CANCELED if sent during C30 or S20 commands.

XPI reference: S00

10.36.2.22 - (void) cancelCommandWithDisplayOptions: (int) *display* andEMVFallback:(int) *fallbackFlag* disableKeyboard:(BOOL) *shouldDisableKeyBoard*

Cancel Command With Display Options and EMV Fallback and Change Keyboard Status.

This command will cancel any PIN pad command that is pending User Action such as:

- Card insertion or card swipe
- Data Entry or Function Key
- Signature Capture Request

Parameters

display	int value to specify what the screen should display after this command • 0: No Screen Change • 1: Display Welcome Message • 2: Clear The Screen
fallbackFlag	int value to specify the EMV Fallback Setting • 0: Set Fallback Flag Off • 1: Set Fallback Flag On Any other value - not sent
shouldDisable-Keyboard	BOOL value to specify the keyboard behavior after the command is processed • false: Keyboard remains unchanged with the current status (default) • true: Disable Keyboard

It is recommended that the host device send an S01 command after the S00 command if the WELCOME/BONJOUR message is to appear. S00 Command cancels both US and Canadian commands. S00 will not display TRANSACTION CANCELED if sent during C30 or S20 commands.

XPI reference: S00

10.36.2.23 - (void) cancelPendingCommand

Clears VMF buffer of any pending command not yet sent to e210/e255/e315 then executes cancelCommand

10.36.2.24 - (void) cancelPendingCommandNoData

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.36.2.25 - (int) cardAuthorization: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray*

EMV Transaction Request.

Authorizes the EMV transaction amounts using the inserted chip (EMV) card. This request should follow the response to a request to getCardData:language:amount:otherAmount: if the response indicates the card was inserted.

If result code returns 0-Successful, then the card data can be read from vfiEMVAuthorization

XPI reference: C32

Parameters

amount	Transaction amount value
amtOther	Other transaction amount value

decision	The merchant decision notes additional handling for the EMV request based on required processor handling. Values are: • 0: Request a TC • 1: Force online – 2: Force decline (AAC)
reqObjArray	A dictionary containing EMV tags of type NSData that is required at this transaction stage to be retrieved from the inserted card. See setEMVTags: for an example of creating a dictionary of EMV tags.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 26: Card Swiped during Fallback mode • 42: Candidate List Empty

10.36.2.26 - (int) cardAuthorization: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray* displayResult:(BOOL) *display*

EMV Transaction Request.

Authorizes the EMV transaction amounts using the inserted chip (EMV) card. This request should follow the response to a request to getCardData:language:amount:otherAmount: if the response indicates the card was inserted.

If result code returns 0-Successful, then the card data can be read from vfiEMVAuthorization

XPI reference: C32

Parameters

amount	Transaction amount value
amtOther	Other transaction amount value
decision	The merchant decision notes additional handling for the EMV request based on required processor handling. Values are: • 0: Request a TC • 1: Force online – 2: Force decline (AAC)
reqObjArray	A dictionary containing EMV tags of type NSData that is required at this transaction stage to be retrieved from the inserted card. See setEMVTags: for an example of creating a dictionary of EMV tags.
display	Display Card Decision • 0: Display decision • 1: Do not display decision

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 26: Card Swiped during Fallback mode • 42: Candidate List Empty

10.36.2.27 - (int) cardAuthorization: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray* displayResult:(BOOL) *display* pinTryExceedStatus:(BOOL) *pinTry* displayAmount:(BOOL) *displayAmt*

EMV Transaction Request.

Authorizes the EMV transaction amounts using the inserted chip (EMV) card. This request should follow the response to a request to getCardData:language:amount:otherAmount: if the response indicates the card was inserted.

If result code returns 0-Successful, then the card data can be read from vfiEMVAuthorization

XPI reference: C32

Parameters

amount	Transaction amount value
amtOther	Other transaction amount value
decision	The merchant decision notes additional handling for the EMV request based on required processor handling. Values are: • 0: Request a TC • 1: Force online – 2: Force decline (AAC)
reqObjArray	A dictionary containing EMV tags of type NSData that is required at this transaction stage to be retrieved from the inserted card. See setEMVTags: for an example of creating a dictionary of EMV tags.
display	Display Card Decision • 0: Display decision • 1: Do not display decision
pinTry	Display PIN try exceeded. Pin Try Exceeds Limit prompt will be display in Offline Pin Entry for cases that the try limit has been exceeded. • 0: Display PIN try exceeded • 1: Do not display PIN try exceeded
displayAmt	Display amount. If enabled, the amount prompt will be shown during Offline Pin Entry. • 0: Display amount • 1: Do not display amount

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 26: Card Swiped during Fallback mode • 42: Candidate List Empty

10.36.2.28 - (int) cardAuthorization: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray* displayResult:(BOOL) *display* pinTryExceedStatus:(BOOL) *pinTry* displayAmount:(BOOL) *displayAmt* displayAppExpired:(BOOL) *displayAppExpired*

EMV Transaction Request.

Authorizes the EMV transaction amounts using the inserted chip (EMV) card. This request should follow the response to a request to getCardData:language:amount:otherAmount: if the response indicates the card was inserted.

If result code returns 0-Successful, then the card data can be read from vfiEMVAuthorization

XPI reference: C32

Parameters

amount	Transaction amount value

amtOther	Other transaction amount value
decision	The merchant decision notes additional handling for the EMV request based on required processor handling. Values are: • 0: Request a TC • 1: Force online – 2: Force decline (AAC)
reqObjArray	A dictionary containing EMV tags of type NSData that is required at this transaction stage to be retrieved from the inserted card. See setEMVTags: for an example of creating a dictionary of EMV tags.
display	Display Card Decision • 0: Display decision • 1: Do not display decision
pinTry	Display PIN try exceeded. Pin Try Exceeds Limit prompt will be display in Offline Pin Entry for cases that the try limit has been exceeded. • 0: Display PIN try exceeded • 1: Do not display PIN try exceeded
displayAmt	Display amount. If enabled, the amount prompt will be shown during Offline Pin Entry. • 0: Display amount • 1: Do not display amount
displayApp-Expired	Display App Expired • 0: Display app expired • 1: Do not display app expired

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 26: Card Swiped during Fallback mode • 42: Candidate List Empty

10.36.2.29 - (void) clearAllData

Clears all data structures in VFIPinpad class.

10.36.2.30 - (int) clearCAPK

Clear all CAPK Files.

This command will scan through the EST and delete all the current CAPK files. XPI reference: D12.

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in Table Record (Future Use) • 09: Table Filename Duplication (Future Use)

10.36.2.31 - (void) closeDevice

Terminates stream connection to XPI application.

This method will shut down the connection to the XPI stream. An initDevice will need to be executed again to engage a new stream connection.

10.36.2.32 - (int) combinedPromptCommand: (float) *amount* language:(int) *lcode* transID:(int) *ID* menuArray:(NSString ∗) *promptArray* surcharge:(float) *surchgAmt* accountNum:(NSString ∗) *account* tipOptions:(NSString ∗) *tipOpt* cashBackOptions:(NSString ∗) *cashBackOpt* surchargeOptions:(int) *surchargeOpt*

Combined Prompt Command.

This command allows the terminal or ECR to send many customer prompt commands and to define the order of those prompts - all with one request to the PIN pad. It includes Enhanced Tip and CashBack options for the customer.

This command also supports the backup of prompts each time the CORR key is pressed.

Prompts can be called in the promptArray. For example, to call the following prompts in this order: Confirm Amount - Tip - Confirm Amount - Select Account - Enter PIN The value of promptArray would be “0105010203”.

Upon successful execution, response can be obtained from property vfiDataEntry.

XPI reference: S23

Parameters

amount	Transaction amount value that is to be displayed
lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
ID	Transaction Name ID.
promptArray	Acceptable Values: • 01: Confirm Transaction Amount • 02: Select Account • 03: Enter (Online)PIN • 04: CashBack • 05: Tip Entry • 06: Surcharge/Fee • 07: Confirm Total Amount For example, to call the following prompts in this order: Confirm Amount - Tip - Confirm Amount - Select Account - Enter PIN The value of promptArray would be “0105010203”.
surchgAmt	An optional surcharge amount that is to be displayed and confirmed (valid range is from 0.01 to 99.99)
account	Account number is checked for the length (greater than 8 and less than 25) and all numeric value.
tipOpt	Refer to Enhanced Tip Entry and Setup
cashBackOpt	Refer to Enhanced CashBack Entry and Setup
surchargeOpt	Acceptable Values • 01: Always Prompt Surcharge • 02: Only prompt if CashBack entered Field Separator

Return values

Result	Code: • 00: Command was successful. Retrive card data from vfiDataEntry • 01: Command was unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed • 06 - Invalid Account Number length • 50 - Invalid Tip Setup • 51 - Invalid Surcharge Setup • 52 - Invalid CashBack Parameters • 79 - Security keys missing

Enhanced Tip Entry and Setup

Tip Entry options are all configurable and defined in the data sent. Options for Tip Entry are as follows:

1. Select a pre-set dollar amount
2. Select a pre-set percentage amount
3. Enter own tip amount

The tip amount entered or selected will be added to the total amount for any following 'Confirm Amount' prompts. However, if the CORR key is pressed, and the Tip Entry process starts again, the amount will be removed from the total amount.

Pressing the CORR key at the next prompt will not return to any sub-screen in the Tip Entry process, but will instead restart the Tip Entry process.

Enhanced Tip Entry Setup - tipOpt parameter

The tipOpt parameter that is sent in to this command is defined as follows:

`<Tip Entry Flag><*><$ Menu Selections><*><% Menu Selections>`

`<Tip Entry Flag>` defines if enhanced Tip is required:

- 0 - Prompt for Enter Tip only - no Enhanced Tip.
- 1 - Enhanced Tip screens to be prompted

Note: If Enhanced Tip is enabled and there are missing "Tip Options" settings, XPI will return error code 50 (Invalid Tip Setup).

`<$ Menu Selections>` is an array of 8 digits to be used for menu selections on the amount selection screen:

- First 2 digits - the first $ value to appear on the menu
- Second 2 digits - the second $ value to appear on the menu
- Third 2 digits - the third $ value to appear on the menu
- Fourth 2 digits - the fourth $ value to appear on the menu

Note: The length of this field is to be 8 digits if Enhanced Tip is defined. Otherwise, this field can be left empty.

`<% Menu Selections>` is an array of 8 digits to be used for menu selections on the percentage selection screen:

- First 2 digits - the first % value to appear on the menu
- Second 2 digits - the second % value to appear on the menu
- Third 2 digits - the third value to appear on the menu%
- Fourth 2 digits - the fourth % value to appear on the menu

  Note: The length of this field is to be 8 digits if Enhanced Tip is defined. Otherwise, this field can be left empty.

  Examples for the "Tip Options" data:

- “1∗10203040∗10203040”, where Enhanced Tip is defined, and the options for both $ tip and % tip selections are 10, 20, 30 and 40.
- “0∗∗”, where only the Enter Tip prompt will appear

Enhanced Tip Entry Setup

Tip Screens

The following screens appear during the Enhanced Tip entry process:

Tip Options Menu∗∗ - This menu will appear if Enhanced Tip has been enabled:

- Selecting “$” will prompt the “$ Selection Menu”
- Selecting “” will prompt the “% Selection Menu”
- Selecting “Other” will prompt the “Enter Tip” prompt
- Selecting “None” will exit the Tip Entry process

  $ Selection Menu∗∗ - This menu will appear if the “$” selection was made in the Tip Options menu:

- Selecting any option in this menu will return the dollar value with two trailing zeros (implied decimal) in the Tip Amount value in the response data.
- Pressing the CORR key will return to the Tip Options menu if previously displayed.
- Note the values 10, 20, 30, and 40 are example data only.

  % Selection Menu∗∗ - This menu will appear, if the “” selection was made in the Tip Options menu:

- Selecting any option in this menu will calculate the % tip to the transaction amount sent in the request, not including any cashback or surcharge amount, and then add the calculated value with two trailing zeros (implied decimal) in the Tip Amount value in the response data.
- Pressing the CORR key will return to the Tip Options menu, if previously displayed.
- Note the values 10, 20, 30 and 40 are example data only.

Enhanced CashBack Entry and Setup

Cashback Entry options are all configurable and defined in the data sent. Options for Cashback are as follows:

1. Select a pre-set dollar amount
2. Enter own cashback amount

The Cashback amount entered or selected will be added to the total amount in any following Confirm Amount prompts. However, if the CORR key is pressed and the CashBack process starts again, the amount must be removed from the total amount.

Pressing the CORR key at the next prompt, will not return to any sub-screen in the CashBack Entry process, but will instead restart the Cashback process.

Enhanced CashBack Entry Setup - cashBackOpt parameter

The cashBackOpt parameter that is sent in to this command is defined as follows:

`<CashBack Entry Flag><*><$ Menu Selections>`

`<CashBack Entry Flag>` defines if Enhanced CashBack is required:

- 0 - Prompt for Enter CashBack only - no Enhanced Cashback.
- 1 - Enhanced CashBack screens to be prompted

Note: If Enhanced Cashback is enabled and there are missing settings sent, then XPI will return error code 51 (Invalid Cashback Setup).

`<$ Menu Selections>` is an array of 8 digits to be used for menu selections on the amount selection screen:

- First 2 digits - the first $ value to appear on the menu
- Second 2 digits - the second $ value to appear on the menu
- Third 2 digits - the third $ value to appear on the menu
- Fourth 2 digits - the fourth $ value to appear on the menu

Note: The length of this field is to be 8 digits if Enhanced CashBack is enabled. Otherwise, this field can be left empty..

Examples for the "Cashback Options" data:

- "1∗10203040" will turn on enhanced CashBack and will display the Enhanced CashBack screens with the options of 10, 20, 30 and 40 appearing if the pre-set $ selection is selected
- "0∗∗", where only the Enter CashBack prompt will appear (no Enhanced CashBack).

Enhanced CashBack Screens

Tip Screens

The following screen appears during the CashBack entry process:

CashBack Options Menu∗∗ - This menu will appear if Enhanced CashBack has been enabled:

- Selecting "$" will prompt the "$ Selection Menu"
- Selecting "Other" will prompt the "Enter CashBack" prompt that is currently supported
- Selecting "None" will exit the CashBack Entry process

10.36.2.33 - (int) completeOnlineEMV: (int) *hostDecision* displayResult:(BOOL) *flag* hostTags:(NSMutableDictionary ∗) *hostTagData*

Complete Online EMV Transaction.

This command is required to complete an online EMV transaction. EMV tags returned in vfiEMVCompletionData emvTags variable

XPI reference: C34

Parameters

hostDecision	indicator from the host response; valid values are:: • 0: Host Authorized • 1: Host Declined • 2: Failed to Connect
flag	Indicator to note whether or not to display the final response message; the valid values are: • 0: Do not display • 1: Display
hostTagData	NSMutableDictionary containing EMV tags. The dictionary has the `<NSString>` for keys representing EMV Tag name, and `<NSString>` for objects representing tag data.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 26: Card Swiped during Fallback mode • 42: Candidate List Empty • 91: Issuer authentication failed, or the card is in an error state (EMV Status Code=0x6985)

Converting TVL to NSMutableDictionary

EMV data is usually received in TLV (Tag, Length, value) format: `950500000080009B02E8009F2701018A025-A339F26080C552B9364D55CE5`

This data contains the following EMV tags/values:

Tag	Length	Value
95	05	0000008000
9B	02	E800
9F27	01	01
8A	02	5A33
9F26	08	0C552B9364D55CE5

An example how to create an NSMutableDictionary with these values follows.

```
-(void) setTagsAndSend{

NSMutableDictionary *emvTags = [[NSMutableDictionary alloc] initWithCapacity:0];

[emvTags setObject:@"0000008000" forKey:@"95"];
[emvTags setObject:@"E800" forKey:@"9B"];
[emvTags setObject:@"01" forKey:@"9F27"];
[emvTags setObject:@"5A33" forKey:@"8A"];
[emvTags setObject:@"0C552B9364D55CE5" forKey:@"9F26"];
[emvTags setObject:@"01" forKey:@"C2"];

[self completeOnlineEMV:0 displayResult:0 hostTags:emvTags];

}
```

10.36.2.34 - (void) consoleEnabled: (BOOL) *enable*

Enabled logging to XCode Console.

Parameters

enable	Setting *TRUE* enables additional logging to debug b_console window in iOS

10.36.2.35 - (NSString *) copyCAPKFile

Filename of newly created CAPK from createCAPK()

CAPK File name format is the first and last letter of the RID plus the Index number. The extension is EMV.

Return values

NSString	of CAPK Created, if successful

See Also

- createCAPK:pkIndex:modulus:exponent:checksum:expiryDate:

10.36.2.36 - (NSDictionary *) copyLastEMVTags

Last EMV tags.

Return values

NSDictionary	last successfully processed TVL stream converted to a dictionary with EMV tag values.

10.36.2.37 - (NSString ∗) copyPromptData

General Prompt Response.

Creates a new NSString object containing the results of the last general prompt request command.

Return values

NSString	object containing the cardholder entered data in response to the general prompt request command task

See Also

- generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry:displayCanceled:restoreScreen:
- generalPromptDefault:message2:minLength:maxLength:language:initialValue:allowSkip:

10.36.2.38 - (NSString ∗) copyStringResponse

Input returned from requestStringInput()

Return values

NSString	of keys (from '0' to '9') corresponding to the key inputs on the e210/e255/e315

See Also

- requestStringInput:timeout:maxLen:

10.36.2.39 - (int) createCAPK: (NSString ∗) *RID* pkIndex:(int) *index* modulus:(NSData ∗) *modulus* exponent:(NSData ∗) *exp* checksum:(NSData ∗) *csum* expiryDate:((deprecated("use method createCAPKFile instead"))) __*attribute*

Create CAPK File.

This command will create the CAPK File and update the EST table. Application will search for the existing CAPK file name in the EST and if no match is found, the new CAPK File name will be inserted in the empty slot.

CAPK File name format is the first and last letter of the RID plus the Index number. The extension is EMV.

Ex: A398.EMV is the CAPK file name for RID A000000003, PK Index 98 for SC 5000 devices. The name will be different for other devices.

If successful, file name of CAPK created can be retrieve by calling copyCAPKFile

XPI reference: D11.

Parameters

RID	RID Number, ASCII string
index	PK Index (ASCII)
modulus	Modulus value (HEX)
exp	Exponent value (HEX)
csum	Hash Value (Hex 20 bytes), Optional
date	Expiry Date, Optional

Return values

Result	Code: • 00: Successful - retrieve file name of CAPK by calling copyCAPKFile • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.40 - (int) createCAPKFile: (NSString ∗) *RID* pkIndex:(NSString ∗) *index* modulus:(NSData ∗) *modulus* exponent:(NSData ∗) *exp* checksum:(NSData ∗) *csum* expiryDate:(NSDate ∗) *date*

Create CAPK File.

This command will create the CAPK File and update the EST table. Application will search for the existing CAPK file name in the EST and if no match is found, the new CAPK File name will be inserted in the empty slot.

CAPK File name format is the first and last letter of the RID plus the Index number. The extension is EMV.

Ex: A398.EMV is the CAPK file name for RID A000000003, PK Index 98 for SC 5000 devices. The name will be different for other devices.

If successful, file name of CAPK created can be retrieve by calling copyCAPKFile

XPI reference: D11.

Parameters

RID	RID Number, ASCII string
index	PK Index (ASCII)
modulus	Modulus value (HEX)
exp	Exponent value (HEX)

csum	Hash Value (Hex 20 bytes), Optional
date	Expiry Date, Optional

Return values

Result	Code: • 00: Successful - retrieve file name of CAPK by calling copyCAPKFile • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.41 - (int) createMac: (NSData *) *macData*

Create Mac Request.

Accepts a string of data to be MAC'ed using the ANSI x9.91 standard and the MAC Working Key. This is used for MAC'ing credit transactions when the PINPad is configured to support both credit and debit.

Upon successful execution, response can be obtained from property vfiEncryptionData.

XPI reference: S66

Parameters

macData	String of data to be MAC'ed

Return values

Result	Code: • 00: Command was successful. Retrive card data from vfiEncryptionData • 01: Command was unsuccessful • 79: No Keys, no data

10.36.2.42 - (int) createTable: (int) *tableID* numRecords:(int) *reccnt*

Create EMV Tables.

This command will create an empty EMV Card Scheme Table (EST), EMV Configuration Table (MVT), Extended EMV Card Scheme Table (EEST) and Extended EMV Configuration Table (EMVT) on the PIN pad.

XPI reference: D16.

Parameters

tableID	Table Type, must be one of the following values: • 01: Create EST Table • 02: Create MVT Table • 03: Create EEST Table • 04: Create EMVT Table
reccnt	Number of records in the designated table to create (001 - 999)

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID. • 10: Invalid table ID .

10.36.2.43 - (NSString∗) ctlsOPTFLAG

OPTFLAG values from statusCTLS

Value populated after running statusCTLS.

Return values

NSString	of 16 Digits representing OPTFLAG

See Also

- statusCTLS

10.36.2.44 - (int) ctlsState

Contactless reader state after running statusCTLS

Value populated after running statusCTLS.

Return values

int	value representing contactless reader state • 00: Contactless Reader not yet initialized • 01: Contactless Reader has been initalized.

See Also

- statusCTLS

10.36.2.45 - (int) D01: (BOOL) *bypass*

Set PIN Bypass Support.

See Also

- pinBypass:

10.36.2.46 - (int) D11: (NSString ∗) *RID* pkIndex:(int) *index* modulus:(NSData ∗) *modulus* exponent:(NSData ∗) *exp* checksum:(NSData ∗) *csum* expiryDate:((deprecated("use method D11a instead"))) __*attribute*

Create CAPK File.

See Also

- createCAPK:pkIndex:modulus:exponent:checksum:expiryDate:

10.36.2.47 - (int) D11a: (NSString ∗) *RID* pkIndex:(NSString ∗) *index* modulus:(NSData ∗) *modulus* exponent:(NSData ∗) *exp* checksum:(NSData ∗) *csum* expiryDate:(NSDate ∗) *date*

Create CAPK File.

See Also

- createCAPKFile:pkIndex:modulus:exponent:checksum:expiryDate:

10.36.2.48 - (int) D12

Clear all CAPK Files.

See Also

- clearCAPK

10.36.2.49 - (int) D13: (int) *recno* AID:(int) *aidNo* partialNameMatch:(BOOL) *flag* supportedAID:(NSData ∗) *supAID* termAVN:(NSData ∗) *avn* term2ndAVN:(NSData ∗) *avn2* recommendedAIDName:(NSString ∗) *aidName*

Update AID in EST (EMV Card Scheme Table)

See Also

- updateAIDRecord:AID:partialNameMatch:supportedAID:termAVN:term2ndAVN:recommendedAIDName:

10.36.2.50 - (int) D14: (int) *recno* tableData:(VFIEMVConfiguration ∗) *emvData*

Update AID in EST (EMV Card Scheme Table)

See Also

- updateEMVConfigRecord:tableData:

10.36.2.51 - (int) D15: (int) *recno* numberTrans:(long) *numTrans* scheme:(NSString ∗) *schemeLabel* regId:(NSString ∗) *RID* CSNList:(NSString ∗) *csn* emvTableRecord:(long) *rec* CSNFile:(NSString ∗) *csnFile*

Update Scheme Record.

See Also

- updateSchemeRecord:numberTrans:scheme:regId:CSNList:emvTableRecord:CSNFile:

10.36.2.52 - (int) D16: (int) *tableID* numRecords:(int) *reccnt*

Create EMV Tables.

See Also

- createTable:numRecords:

10.36.2.53 - (int) D17: (int) *recno* emvRecord:(int) *emvRecno* AID:(NSString ∗) *aid* maxAIDLen:(int) *maxLen* appFlow:(int) *flow* partialName:(BOOL) *pflag* acctSelect:(BOOL) *aflag* enableCtls:(BOOL) *cflag* transScheme:(int) *scheme*

Update EEST Table.

See Also

- updateExtendedSchemeRecord:emvRecord:AID:maxAIDLen:appFlow:partialName:acctSelect:enableCtls-:transScheme:

10.36.2.54 - (int) D18: (NSData ∗) *termCAPQPS* sig:(NSData ∗) *termCAPSig* CVM:(float) *cvmAmount*

Initialize CTLS Reader.

See Also

- initializeContactless:sig:CVM:

10.36.2.55 - (int) D19: (int) *recno* mvtIndex:(int) *emvIndex* groupName:(NSString ∗) *group* ctlsFloorLimit:(float) *ctlsFloor* ctlsCVMLimit:(float) *ctlsCVM* ctlsTranLimit:(float) *ctlsTran* ctlsTacDefault:(NSData ∗) *tacDefault* ctlsTacDenial:(NSData ∗) *tacDenial* ctlsTacOnline:(NSData ∗) *tacOnline* visaTTQ:(NSData ∗) *ttq*

Update EMVT.

See Also

- updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:

10.36.2.56 - (int) D19: (int) *recno* mvtIndex:(int) *emvIndex* groupName:(NSString ∗) *group* ctlsFloorLimit:(float) *ctlsFloor* ctlsCVMLimit:(float) *ctlsCVM* ctlsTranLimit:(float) *ctlsTran* ctlsTacDefault:(NSData ∗) *tacDefault* ctlsTacDenial:(NSData ∗) *tacDenial* ctlsTacOnline:(NSData ∗) *tacOnline* visaTTQ:(NSData ∗) *ttq* termCapabilities:(NSString ∗) *termCapb* addTermCapabilities:(NSString ∗) *addTermCapb*

Update EMVT.

See Also

- updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:

10.36.2.57 - (int) D19: (int) *recno* mvtIndex:(int) *emvIndex* groupName:(NSString ∗) *group* ctlsFloorLimit:(float) *ctlsFloor* ctlsCVMLimit:(float) *ctlsCVM* ctlsTranLimit:(float) *ctlsTran* ctlsTacDefault:(NSData ∗) *tacDefault* ctlsTacDenial:(NSData ∗) *tacDenial* ctlsTacOnline:(NSData ∗) *tacOnline* visaTTQ:(NSData ∗) *ttq* termCapabilities:(NSString ∗) *termCapb* addTermCapabilities:(NSString ∗) *addTermCapb* ctlsCountry:(NSString ∗) *country* ctlsCurrency:(NSString ∗) *currency*

Update EMVT.

See Also

- updateExtendedEMVRecord:mvtIndex:groupName:ctlsFloorLimit:ctlsCVMLimit:ctlsTranLimit:ctlsTacDefault:ctlsTacDenial:ctlsTacOnline:visaTTQ:termCapabilities:addTermCapabilities:

10.36.2.58 - (int) D20: (int) *recno* ctlsMerchantType:(int) *merchType* ctlsTermTransInfo:(NSString ∗) *transInfo* ctlsTermTransType:(NSString ∗) *transType* ctlsReqReceiptLimit:(float) *ctlsReceipt* ctlsOptionStatus:(NSString ∗) *optionStatus* ctlsReaderFlooorLimit:(float) *ctlsFloor*

Update ICCT.

See Also

- updateICCT:ctlsMerchantType:ctlsTermTransInfo:ctlsTermTransType:ctlsReqReceiptLimit:ctlsOptionStatus:ctlsReaderFlooorLimit:

10.36.2.59 - (int) D25: (BOOL) *visaDebitFlag* ctlsMode:(int) *mode* pdolSupport:(BOOL) *pdolFlag*

Additional EMV Info.

See Also

- setAdditionalEVMFlags:ctlsMode:pdolSupport:

10.36.2.60 - (int) D25: (BOOL) *visaDebitFlag* ctlsMode:(int) *mode* pdolSupport:(BOOL) *pdolFlag* valueLinkCtls:(BOOL) *valuLinkFlag* amexMode:(int) *amexMode*

Additional EMV Info.

See Also

- setAdditionalEVMFlags:ctlsMode:pdolSupport:

10.36.2.61 - (int) D26: (int) *recordNum* aid:(NSString ∗) *aid* version:(NSString ∗) *version*

Update PassThru Configuration File.

See Also

- updatePassThru:aid:version:

10.36.2.62 - (int) D27

Retrieve Status of Contactless Reader.

See Also

- statusCTLS

10.36.2.63 - (int) D30: (NSString ∗) *line1* line2:(NSString ∗) *line2* line3:(NSString ∗) *line3* line4:(NSString ∗) *line4* amt1:(BOOL) *amt1* amt2:(BOOL) *amt2* amt3:(BOOL) *amt3* amt4:(BOOL) *amt4*

Prompt Message Changes.

See Also

setC30CustomPrompts:line2:line3:line4:line4amt1:amt2:amt3:amt4:amt4:()

10.36.2.64 - (int) dataEntryRequest: (float) *amount* language:(int) *lcode* tip:(BOOL) *tipFlag* cashback:(BOOL) *cashbackFlag* surcharge:(float) *surchgAmt*

Data Entry Request.

Prompts the cardholder for additional information for the transaction based on the PINPad configuration. The additional requested data can include Tip, Cash Back, and Surcharge. The PINPad will prompt the cardholder for this information (if required) and return the collected data.

This command supports backward compatibility. The new enhanced command combinedPromptCommand:language-:transID:(menuArray:surcharge:accountNum:ipOptions:cashBackOptions:surchargeOptions:() is recommended instead when possible.

Upon successful execution, response can be obtained from property vfiDataEntry.

XPI reference: S21

Parameters

amount	Transaction amount value that is to be displayed
lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
tipFlag	Indicates whether or not to prompt for tip and collect the amount entered.
cashbackFlag	Indicates whether or not to prompt for cash back amount and collect the amount entered
surchgAmt	An optional surcharge amount that is to be displayed and confirmed (valid range is from 0.01 to 99.99). A zero amount will cause this prompt to be bypassed.

Return values

Result	Code: • 00: Command was successful. Retrive card data from vfiDataEntry • 01: Command was unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed

Deprecated Use combinedPromptCommand:language:transID:(menuArray:surcharge:accountNum:ipOptions:cashBackOptions:surchargeOptions:()

10.36.2.65 - (int) dataEntryRequestPrompt: (NSString *) *prompt*

Data Entry Request.

This command is sent to prompt for Tip, Cashback, and Surcharge at the PIN pad. The PIN pad will prompt the customer to enter the information (if required) and send the data collected to the terminal. This command supports backward compatibility. The new enhanced command combinedPromptCommand:language:transID:(menuArray:surcharge:accountNum:ipOptions:cashBackOptions:surchargeOptions:() is recommended instead when possible.

Upon successful execution, response can be obtained from property vfiDataEntry tipAmount.

XPI reference: S21

Parameters

prompt	Secure message prompt (.e.g. ENTER TIP).

Return values

Result	Code: • -01: Framework timeout reached • 00: Command was successful. Retrive card data from vfiDataEntry tipAmount. • 01: Command was unsuccessful • 03: No response: Either CANCEL was pressed on keypad, or XPI timeout reached • 04: CORR key was pressed

Deprecated Use combinedPromptCommand:language:transID:(menuArray:surcharge:accountNum:ipOptions:cashBackOptions:surchargeOptions:()

10.36.2.66 - (int) derivedKey: (int) *command*

Derived Key.

XPI 6.4.2 supports VSP Derived Key. XPI supports a new command that replaces the requirement to use Registart cards at deployment. In addition, the new command can be used to Advance the key. The Key Advance will not be used often since the key is already changed on every transaction. The E20 Command was added in XPI 6.4.1 release. However, XPI 6.4.1 was not a general release. Therefore, the description of the change for Derived Key is included in the XPI 6.4.2 release notes. The Encryption can be stopped but this would only be used if the merchant is no longer using the VSP service.

Upon successful execution, MSR Synchronization data can be obtained from track property vfiCardData (optional Track 1, 2, 3).

XPI reference: D20

Parameters

Command	Enable, Disable and Advance VSP Derived Key Feature • 00: Registart • 01: Registop • 02: Advance DDK

Return values

Result	Code: • 00: Command was successful. Retrive MSR Synchronization data from vfiCardData (optional Track 1, 2, 3) • 01: Command was unsuccessful

10.36.2.67 - (int) diagnosticInfo

Polls e210/e255/e315 for Diagnostic Info.

This will request diagnostic info from the e210/e255/e315. The returned data will be populated in the VFIDiagnostics object vfiDiagnostics. XPI reference: S95.

Return values

Result	Code: • 00: Successful • 01: Unsuccessful

10.36.2.68 - (int) diagnosticInfoExtended

Polls e210/e255/e315 for Diagnostic Info.

This will request diagnostic info from the e210/e255/e315. Includes polling for extended data SVC_INFO_SERLNO and SVC_INFO_PARTNO. The returned data will be populated in the VFIDiagnostics object vfiDiagnostics. XPI reference: S95.

Return values

Result	Code: • 00: Successful • 01: Unsuccessful

10.36.2.69 - (void) disableBlocking

Disables default blocking on all API methods.

By default blocking is on. Most API calls will wait for a response from XPI before returning control to integrator. This behavior can be turned off by calling this method. Default blocking can be turned back on by calling enableBlocking

10.36.2.70 - (void) disableMSR

Toggle the Card Reader Off.

This command disallows data entry from the card reader without affecting the PINPad display. XPI reference: Q41

10.36.2.71 - (void) displayMessage: (NSString *) *string* clearScreen:(BOOL) *clear*

Display a String.

This command will display a message on the Pinpad. The Pinpad displays the message until the customer presses the 'CANCEL' key, or until it receives another packet that changes the message content or forces it back to the idle state.

This command uses secure messages which must be defined within XPI's available message list in order for this method to generate the message on the screen.

Parameters

string	Secure message to display
clear	Clear Display

- Secure Message Table

XPI reference: Z2

Deprecated Use displayMessages:message1:message2:clearScreen:

10.36.2.72 - (int) displayMessageAndWait: (NSString *) *msg1* line2:(NSString *) *msg2*

Display Message and Wait for Key.

This command is used to display message on the external Pinpad until a key is pressed. Valid keys are <OK>, <CORR>, or <CANCEL> keys. This command is used for card holder acceptance input. Timeout is set to 15 seconds.

XPI reference: S11

Parameters

msg1	A message string to display on the PINPad screen. The message may consist of up to 16 characters.
msg2	A message string to display on the second line of the PINPad screen. The message may consist of up to 16 characters

Return values

Result	Code: • 00000: OK Key pressed • 00001: CORR Key pressed • 00002: CANCEL Key pressed • 00004: Timeout

NOTE: PIN pad will respond with 00002 code, when request message is more than 16 characters.

10.36.2.73 - (int) displayMessageAndWait: (NSString *) *msg1* line2:(NSString *) *msg2* line3:(NSString *) *msg3* line4:(NSString *) *msg4*

Display Message and Wait for Key.

This command is used to display message on the external Pinpad until a key is pressed. Valid keys are <OK>, <CORR>, or <CANCEL> keys. This command is used for card holder acceptance input. Timeout is set to 15 seconds.

XPI reference: S11

Parameters

msg1	A message string to display on the PINPad screen. The message may consist of up to 16 characters.
msg2	A message string to display on the second line of the PINPad screen. The message may consist of up to 16 characters
msg3	A message string to display on the third line of the PINPad screen. The message may consist of up to 16 characters
msg4	A message string to display on the fourth line of the PINPad screen. The message may consist of up to 16 characters

Return values

Result	Code: • 00000: OK Key pressed • 00001: CORR Key pressed • 00002: CANCEL Key pressed • 00004: Timeout

NOTE: PIN pad will respond with 00002 code, when request message is more than 16 characters.

10.36.2.74 - (int) displayMessagePromptInput: (NSString ∗) *message* echo:(int) *echoFlag* timeout:(int) *timeout* maxLen:(int) *length* entryType:(int) *type*

Display Message and Prompt for Input.

This command displays a message and then requests a string of key input from the PIN pad keyboard.

After the string of keys (from '0' to '9') is entered, the PIN pad sends a string of ASCII key values corresponding to the key inputs to the copyStringResponse. The PIN pad may echo the input on its display. If no key input is received within the timeout period, the PIN pad will return to its idle state.

XPI reference: Z52.

Parameters

message	Text message to be displayed.
echoFlag	Indicates whether to echo key input with an asterisk '∗', keyed input, or none at all • 0: Echo key input with '∗' • 1: Echo key input • 2: Don't echo key input
timeout	Sets the input timeout in seconds (0-255)
length	Maximum length of the input string allowed (1-49, 49 is the default)
type	Entry type. Valid values are: • 0= Alphanumeric (Default) • 1= Numeric

Return values

Result	Code: • 01: Success - Retrive value by calling copyStringResponse • 02: Timeout. • 03: NULL. Enter key was pressed without any other input

10.36.2.75 - (void) displayMessages: (NSString ∗) *line1* Line2:(NSString ∗) *line2* Line3:(NSString ∗) *line3* Line4:(NSString ∗) *line4*

Display Messages.

This command will display messages passed by the terminal. This API does not support keyboard input. The PIN pad displays the message until the customer presses the CLEAR key, or until it receives another packet to change the message content or force it back to idle state.

XPI reference: S14

Parameters

line1	A message to display on line 1 of the PINPad screen. May consist of up to 16 characters.
line2	A message to display on line 2 of the PINPad screen. May consist of up to 16 characters.
line3	A message to display on line 3 of the PINPad screen. May consist of up to 16 characters.
line4	A message to display on line 4 of the PINPad screen. May consist of up to 16 characters.

Additional Information:

- Sixty (60) seconds is the timeout default.
- All display messages are aligned to the center of the screen.
- Message lines support a maximum of 16 displayable characters. Messages greater than 16 characters will not fully display or will display blank.

10.36.2.76 - (void) displayMessages: (NSString ∗) *line1* Line2:(NSString ∗) *line2* Line3:(NSString ∗) *line3* Line4:(NSString ∗) *line4* timeout:(int) *timeout*

Display Messages.

This command will display messages passed by the terminal. This API does not support keyboard input. The PIN pad displays the message until the customer presses the CLEAR key, or until it receives another packet to change the message content or force it back to idle state.

XPI reference: S14

Parameters

line1	A message to display on line 1 of the PINPad screen. May consist of up to 16 characters.
line2	A message to display on line 2 of the PINPad screen. May consist of up to 16 characters.
line3	A message to display on line 3 of the PINPad screen. May consist of up to 16 characters.
line4	A message to display on line 4 of the PINPad screen. May consist of up to 16 characters.
timeout	Timeout value. 99999 = No Timeout

Additional Information:

- Sixty (60) seconds is the timeout default.
- All display messages are aligned to the center of the screen.
- Message lines support a maximum of 16 displayable characters. Messages greater than 16 characters will not fully display or will display blank.

10.36.2.77 - (void) displayMessages: (int) *numMessages* message1:(NSString ∗) *msg1* message2:(NSString ∗) *msg2* clearScreen:(BOOL) *clear*

Display Messages.

This command will display up to 2 messages on the Pinpad. The Pinpad displays the message until the customer presses the 'CANCEL' key, or until it receives another packet that changes the message content or forces it back to the idle state.

This command uses secure messages which must be defined within XPI's available message list in order for this method to generate the message on the screen.

Parameters

numMessages	Number of messages to display. Valid values 0 - 2
msg1	First secure message to display
msg2	Second secure message to display
clear	NOT APPLICABLE: XPI does not support Clear Screen on e210/e255/e315. Parameter ignored

- Secure Message Table

XPI reference: Z3

10.36.2.78 - (int) doInteracRequest: (int) *transId* language:(int) *lcode* amount:(float) *amt* pan:(NSString ∗) *acctNumber* mac:(NSData ∗) *macData*

Perform Interac Request Transaction.

Sends a series of data to the PINPad in order to perform a transaction request cycle. The PINPad application will prompt the card holder for confirmation of the total amount, select an account, and PIN entry if required.

Upon successful execution, response can be obtained from property vfiEncryptionData.

XPI reference: S70

Parameters

transId	Transaction type identifier for determining prompts and sequence; valid values are: • 0 – Purchase • 1 – Refund • 2 – Balance Inquiry (not used) • 3 – Purchase Correction • 4 – Refund Correction
lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
amt	Transaction amount that is to be displayed (can be in the range of 0.01 – 999999.99).
acctNumber	Account number with a valid length (greater than 8 and less than 25 characters) and all numeric value.
macData	String of data for MAC generation • 12 characters - PNS Merchant ID • 3 characters - PNS Terminal ID • 2 characters – Transaction Code • PAN Number • Amount (including the decimal) • 6 Digits – Sequence Number • ∗D – Indicate that Account Type will be added to the MAC string at this location, before the MAC calculation. NOTE: The MAC data string shown here as an example only. It should be formatted as required by the host.

Return values

Result	Code: • 00: Command was successful. Retrive card data from vfiEncryptionData • 01: Command was unsuccessful • 02: Time-out occurred, no data • 03: CANCEL key was pressed, no data • 04: CORR key was pressed, no data • 06: Invalid Account Number Length • 79: No Keys, no data

10.36.2.79 - (EncryptionMode) E00

Get Encryption Mode.

See Also

- getEncryptionMode

10.36.2.80 - (int) E02

Retrieve EParms Data.

See Also

- retrieveEParmsData

10.36.2.81 - (int) E04

Perform TGK Update Request.

See Also

- performTGKUpdateRequest

10.36.2.82 - (int) E06

Ge Ciphered Data.

See Also

- getCipheredData

10.36.2.83 - (int) E08_RSA: (NSString *) *publicKeyRSA* publicKeyID:(NSString *) *keyID*

Load Key Information.

See Also

- loadKeyInformation_RSA:publicKeyID:

10.36.2.84 - (int) E08_x509: (NSString *) *publicKeyX509*

Load Key Information.

See Also

- loadKeyInformation_x509:

10.36.2.85 - (EncryptionMode) E10: (EncryptionMode) *mode*

Select Encryption Mode.

See Also

- selectEncryptionMode:

10.36.2.86 - (EncryptionMode) E10: (EncryptionMode) *mode* keyIndex:(int) *index*

Select Encryption Mode.

See Also

- selectEncryptionMode:keyIndex:

10.36.2.87 - (int) E12: (int) *type* p75Data:(NSData *) *data*

Load PKI Signature Files.

See Also

- loadPKISig:p75Data:

10.36.2.88 - (int) E14

Get Key ID Command.

See Also

- getKeyID

10.36.2.89 - (int) E20: (int) *command*

Derived Key.

See Also

- derivedKey:

10.36.2.90 - (void) enableBlocking

Renables default blocking on all API methods.

By default blocking is on. Most API calls will wait for a response from XPI before returning control to integrator. This behavior can be turned off by calling disableBlocking

10.36.2.91 - (void) enableMSR

Toggle the Card Reader On.

This command allows data entry from the card reader without affecting the PINPad display. XPI will only return a response on a successful card read to pinpadMSRData:expMonth:expYear:trackData: (VFIPinpadDelegate-p). If a card read fails, XPI will display CARD READ ERROR and re-prompt without returning a response to the calling application. XPI reference: Q40

Deprecated Replaced by enableMSRDualTrack

10.36.2.92 - (void) enableMSRDualTrack

Toggle the Card Reader On.

This command allows data entry from the card reader without affecting the PINPad display. XPI will only return a response on a successful card read to pinpadMSRData:expMonth:expYear:track1Data:track2Data: (VFIPinpadDelegate-p). If a card read fails, XPI will display CARD READ ERROR and re-prompt without returning a response to the calling application. XPI reference: Q42

10.36.2.93 - (NSString∗) frameworkVersion

Reports current framework version.

10.36.2.94 - (int) generalPrompt: (NSString ∗) *msgID1* message2:(NSString ∗) *msgID2* minLength:(int) *minLen* maxLength:(int) *maxLen* language:(int) *lcode* allowSkip:(BOOL) *flag* secureEntry:(BOOL) *secure*

General Prompt Entry.

Displays prompts that are pre-defined in XPI (e.g. fleet prompts) using the optional protected mode. Each message ID consists of three characters in the range of 000-999. Message IDs correspond to the index in the General Message Table:

- General Message Table

Successful prompt entry will be returned in copyPromptData

XPI reference: S12

Parameters

msgID1	Message ID for line 1 to identify the pre-defined message(s) to display on the PINPad screen.
msgID2	Message ID for line 2 to identify the pre-defined message(s) to display on the PINPad screen.
minLen	Numeric value indicating the minimum data entry length allowed for the prompt response.
maxLen	Numeric value indicating the maximum data entry length allowed for the prompt response.
lcode	Numeric value denoting the language code for determining the displayed language of the pre-defined prompt; NOTE: Any non-zero value will default to French.
flag	Boolean value indicating whether the user can press the enter key to skip the data entry.
secure	Boolean value indicating whether the user entered data should be masked (∗) or not.

Return values

Result	Code: • 00: OK Key pressed. Input data returned in copyPromptData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed

10.36.2.95 - (int) generalPrompt: (NSString ∗) *msgID1* message2:(NSString ∗) *msgID2* minLength:(int) *minLen* maxLength:(int) *maxLen* language:(int) *lcode* allowSkip:(BOOL) *flag* secureEntry:(BOOL) *secure* displayCanceled:(BOOL) *display* restoreScreen:(BOOL) *restore*

General Prompt Entry.

Displays prompts that are pre-defined in XPI (e.g. fleet prompts) using the optional protected mode. Each message ID consists of three characters in the range of 000-999. Message IDs correspond to the index in the General Message Table:

- General Message Table

Successful prompt entry will be returned in copyPromptData

XPI reference: S12

Parameters

msgID1	Message ID for line 1 to identify the pre-defined message(s) to display on the PINPad screen.
msgID2	Message ID for line 2 to identify the pre-defined message(s) to display on the PINPad screen.
minLen	Numeric value indicating the minimum data entry length allowed for the prompt response.
maxLen	Numeric value indicating the maximum data entry length allowed for the prompt response.
lcode	Numeric value denoting the language code for determining the displayed language of the pre-defined prompt; NOTE: Any non-zero value will default to French.
flag	Boolean value indicating whether the user can press the enter key to skip the data entry.
secure	Boolean value indicating whether the user entered data should be masked (∗) or not.

display	Boolean value indicating whether to display CANCELLED when cancelled is pressed not.
restore	Boolean value indicating whether to display idle screen after execution or not.

Return values

Result	Code: • 00: OK Key pressed. Input data returned in copyPromptData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed

10.36.2.96 - (int) generalPromptDefault: (NSString ∗) *msgID1* message2:(NSString ∗) *msgID2* minLength:(int) *minLen* maxLength:(int) *maxLen* language:(int) *lcode* initialValue:(NSString ∗) *defaultValue* allowSkip:(BOOL) *flag*

General Prompt Entry.

Displays prompts that are pre-defined in XPI (e.g. fleet prompts) using the non-protected mode. Each message ID consists of three characters in the range of 000-999. Message IDs correspond to the index in the General Message Table :

- General Message Table

Successful prompt entry will be returned in copyPromptData

This method provides a means to populate the result data with an initial default value. XPI reference: S12

Parameters

msgID1	Message ID for line 1 to identify the pre-defined message(s) to display on the PINPad screen.
msgID2	Message ID for line 2 to identify the pre-defined message(s) to display on the PINPad screen.
minLen	Numeric value indicating the minimum data entry length allowed for the prompt response.
maxLen	Numeric value indicating the maximum data entry length allowed for the prompt response.
lcode	Numeric value denoting the language code for determining the displayed language of the pre-defined prompt; NOTE: Any non-zero value will default to French.
defaultValue	Default value to be used to populate the cardholder data entry initial value
flag	Boolean value indicating whether the user can press the enter key to skip the data entry.

Return values

Result	Code: • 00: OK Key pressed. Input data returned in copyPromptData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed

10.36.2.97 - (int) getCardData: (int) *timeout* language:(int) *lcode* amount:(float) *txnAmt* otherAmount:(float) *txnOthAmt*

Write to Script File.

Prompts the user for card swipe, tap or insert on the PINPad screen and awaits for a designated timeout period for the card data to be read based on the entry method used.

If result code returns 0-Successful, then the card data can be read from vfiCardData Note: When inserting a dual interface card (one that supports both contact and contactless), it is possible for the contactless reader to pick up the contactless signal while the card is read by the contact reader. In the case that both a contact and contactless card read occurs, the contact read will have priority. The contactless read will not be processed.

XPI reference: C30

Parameters

timeout	A timeout value representing the amount of time, in seconds that the customer has to swipe tap, or insert their card.
lcode	Numeric value denoting the language code for determining the displayed language of the pre-defined prompt; NOTE: Any non-zero value will default to French.
txnAmt	Amount for the transaction; this value is required to enable EMV contactless transactions.
txnOthAmt	Any additional amount information for the transaction; this value is only used if EMV Contactless is supported.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 42: Candidate List Empty • 88: No encryption module is activated

10.36.2.98 - (int) getCardData: (int) *timeout* language:(int) *lcode* amount:(float) *txnAmt* otherAmount:(float) *txnOthAmt* isRefundCTLS:(BOOL) *isRefund* disableCTLS:(BOOL) *disableCTLS*

Get Card Data.

Prompts the user for card swipe, tap or insert on the PINPad screen and awaits for a designated timeout period for the card data to be read based on the entry method used.

If result code returns 0-Successful, then the card data can be read from vfiCardData

Note: When inserting a dual interface card (one that supports both contact and contactless), it is possible for the contactless reader to pick up the contactless signal while the card is read by the contact reader. In the case that both a contact and contactless card read occurs, the contact read will have priority. The contactless read will not be processed.

XPI reference: C30

Parameters

timeout	A timeout value representing the amount of time, in seconds that the customer has to swipe tap, or insert their card.
lcode	Numeric value denoting the language code for determining the displayed language of the pre-defined prompt; NOTE: Any non-zero value will default to French.
txnAmt	Amount for the transaction; this value is required to enable EMV contactless transactions.
txnOthAmt	Any additional amount information for the transaction; this value is only used if EMV Contactless is supported.
isRefund	For CTLS, if transaction type is a REFUND, set to TRUE. Otherwise, FALSE = SALE.
disableCTLS	Disabled CTLS reader.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 42: Candidate List Empty • 88: No encryption module is activated

10.36.2.99 - (int) getCipheredData

Get Ciphered Data.

This command is used to retrieve the KeyID and the ciphered Track 2, Track 1, or PAN-Expiry data.

Upon successful response, data can be retrieved in vfiCipheredData keyID, dataType and encryptedBlob values

XPI reference: E06

Return values

Result	Code: • 00: Successful. keyID, dataType and encryptedBlob values from vfiCipheredData • 01: No Encrypted Track Data. • 02: Crypto Error • 99: Command currently Not Supported

10.36.2.100 - (int) getEMVTag: (NSString ∗) *tag*

Get EMV Tag Information.

This command is used to set tag information from the inserted card. The request can get one tag. The result can be retrieved in emvTagData

XPI reference: C20

Parameters

tag	4 character alphanumeric EMV tag in hexadecimal format

Return values

Result	Code: • 00: Successful. Read emvTagData for tag data • 01: Unsuccessful

10.36.2.101 - (int) getEMVTags: (NSMutableArray ∗) *tagArray*

EMV Transaction Request.

This command is used to retrieve tag information from the inserted card. The request can retrieve one or more tags. It is recommended to request all tags in one request.

This command can be used to get multiple EMV Tags from the inserted card. It is recommended to request all desired tags in one request. If result code returns 0-Successful, then the tags can be read from vfiEMVTags

XPI reference: C36

Parameters

tagArray	An array containing EMV tags of type NSData to be retrieved from the inserted card.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error

10.36.2.102 - (EncryptionMode) getEncryptionMode

Get Encryption Mode.

This command can be used to vfi_query XPI for the current encryption mode. XPI reference: E00

Return values

EncryptionMode()	

10.36.2.103 - (int) getKeyID

Get Key ID Command.

This command will return the key ID for the public PKI key currently loaded in the device. If the command is successful, the current keyID is stored in the keyID property of VFICipheredData.

Return values

Result	returns an Integer value • 99: Command Failed/Command Does Not Exist • 0: Command Sucessful

10.36.2.104 - (NSString*) getOSVersion

M01 returns OS version.

10.36.2.105 - (int) getPKICipheredData

Get PKI Ciphered Data.

This command is used to retrieve the KeyID and the ciphered Track 2, Track 1, or PAN-Expiry data.

Upon successful response, data can be retrieved in vfiCipheredData keyID, dataType and encryptedBlob values

XPI reference: E06

Return values

Result	Code: • 00: Successful. keyID, dataType and encryptedBlob values from vfiCipheredData • 01: No Encrypted Track Data. • 02: Crypto Error • 99: Command currently Not Supported

10.36.2.106 - (NSString*) getXPIVersion

M01 returns XPI Version.

10.36.2.107 - (int) I02

Card Remove Request.

See Also

- removeCard

10.36.2.108 - (void) ignoreDisconnect

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.36.2.109 - (id) init

Creates an instance of VFIPinpad class.

Return values

<*id*>	of VFIPinpad class

Example Usage:

```
VFIPinpad* pinpad = [[VFIPinpad alloc] init];
```

10.36.2.110 - (void) initDevice

Initializes the XPI application.

This is executed after the instance is created with init. If any of the optional protocols will be used, setDelegate should first be executed.

Example Usage:

```
VFIPinpad* pinpad = [[VFIPinpad alloc] init];
[pinpad setDelegate:self];
[pinpad initDevice];
```

10.36.2.111 - (void) initDeviceOnServer: (NSString ∗) *address* port:(int) *port*

10.36.2.112 - (int) initializeContactless: (NSData ∗) *termCAPQPS* sig:(NSData ∗) *termCAPSig* CVM:(float) *cvmAmount*

Initialize CTLS Reader.

This command will initialize the contactless reader with the supported Contactless AID's and CAPK files. Each time the AID's or CAPK files are created or changed, this command must be sent to update the reader.

This command is for EMV Contactless only; it is not supported for MSD Contactless.

Either all the optional fields are included or none of them are included. If optional fields are not included, PIN pad does not check TTQ in the config file for CVM setting.

XPI reference: D18.

Parameters

termCAPQPS	Optional, 6-bytes padded with zeros
termCAPSig	Optional, 6-bytes padded with zeros
cvmAmount	CVM Amount; can be up to a 12 digit value including a 2-digit decimal

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.113 - (void) initializeIdle

Query and Initialize Idle Prompt.

This command verifies the existence of the PIN pad and/or initializes the PIN pad's idle prompt. Upon receiving command, e210/e255/e315 will disply the idle prompt. XPI reference: S01.

10.36.2.114 - (void) initServerOnPort: (int) *port*

Initializes the XPI application server.

This is executed after the instance is created with init and the device is initialized. If any of the optional protocols will be used, setDelegate should first be executed.

Parameters

port	Port to listen on

Example Usage:

```
 VFIPinpad* pinpad = [[VFIPinpad alloc] init];
 [pinpad setDelegate:self];
[pinpad initDevice];
 [pinpad initServerOnPort:8080];
```

10.36.2.115 + (BOOL) isGen3Server

10.36.2.116 - (int) loadKeyInformation_RSA: (NSString ∗) *publicKeyRSA* publicKeyID:(NSString ∗) *keyID*

Get PKI Ciphered Data.

The E08 command can be used to load the initial PKI Public Key, or update the existing key. If the E08 command is sent while VSP is enabled, the E10 command will need to be sent to disable VSP and enable PKI encryption.

Upon unsuccessful response, error code can be retrieved in keyLoadErrorCode

XPI reference: E08

Parameters

publicKeyRSA	RSA Publick Key in Base 64 format
keyID	Key Identifer

Return values

Result	Code: • 00: Successful. • 01: Failure. Refer to keyLoadErrorCode

10.36.2.117 - (int) loadKeyInformation_x509: (NSString ∗) *publicKeyX509*

Get PKI Ciphered Data.

The E08 command can be used to load the initial PKI Public Key, or update the existing key. If the E08 command is sent while VSP is enabled, the E10 command will need to be sent to disable VSP and enable PKI encryption.

Upon unsuccessful response, error code can be retrieved in keyLoadErrorCode

XPI reference: E08

Parameters

publicKeyX509	X.509 Public Key Cert in Base 64

Return values

Result	Code: • 00: Successful. • 01: Failure. Refer to keyLoadErrorCode

10.36.2.118 - (int) loadPKISig: (int) *type* p75Data:(NSData ∗) *data*

Load PKI Signature Files.

This command will be used to inject the P7S binary data associated to the “PKI Public Key File” and “Key ID File”. This command must be executed prior to the execution of E08 command.

XPI reference: E12

Parameters

type	Sign File Data associated to: • 00: P7S Data is for PKI Public Key File. • 01: P7S Data is for Key ID File

data	Hex data of the P7S file associated with the Public Key or Key ID Files

Return values

Result	Code: • 00: Successful. • 01: Failure.

10.36.2.119 - (void) logEnabled: (BOOL) *enable*

Enabled logging to delegate.

Parameters

enable	Setting *TRUE* enables logging to pinpadLogEntry:withSeverity: (VFIPinpadDelegate-p)

10.36.2.120 - (int) menuChoiceSelection: (NSString ∗) *msg1* message2:(NSString ∗) *msg2* choice1:(NSString ∗) *cho1* choice2:(NSString ∗) *cho2* choice3:(NSString ∗) *cho3* choice4:(NSString ∗) *cho4*

Display Menu.

This command is used to display a menu on PIN pad devices with function keys that are available beside the screen. For PIN pad devices without function key buttons beside the device screen, the PIN pad application displays numbers corresponding to the function key buttons on the rightmost corner of the device screen. The user should press the corresponding Alpha/Numeric keypad to select an option displayed on the device screen.

After execution of S13, idle screen will not be displayed and will only display a blank screen.

XPI reference: S13

Parameters

msg1	Message Line 1 to display on the PINPad screen; message may consist of up to 16 characters.
msg2	Message Line 2 to display on the PINPad screen; message may consist of up to 16 characters.
cho1	Choice text for selection from the PINPad using the function keys; choice string can consist of up to 16 characters.
cho2	Choice text for selection from the PINPad using the function keys; choice string can consist of up to 16 characters.
cho3	Choice text for selection from the PINPad using the function keys; choice string can consist of up to 16 characters.
cho4	Choice text for selection from the PINPad using the function keys; choice string can consist of up to 16 characters.

Return values

Result	Code: • 00: OK Key pressed • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed • 10: Function Key 1 pressed • 11: Function Key 2 pressed • 12: Function Key 3 pressed • 13: Function Key 4 pressed

On the e210/e255/e315:

- The F1 key is replaced by the '1' key.
- The F2 key is replaced by the '2' key.
- The F3 key is replaced by the '3' key.
- The F4 key is replaced by the '4' key.

Display example:

PIN Pad Display	.	.
.	Choice1	1
Msg Line1	Choice2	2
Msg Line2	Choice3	3
.	Choice4	4

10.36.2.121 - (void) noResponseNextCommand

10.36.2.122 - (int) obtainCardData: (int) *lcode*

Obtain Card Data (Manual Entry or Swipe)

This command prompts the customer for card entry ("ACCOUNT NUMBER / NO COMPTE") and returns the track data or appropriate response code if the card is swiped. If VSP is enabled, the track data will be encrypted by the VSP module.

If card number is manually entered, the entry will be followed by the expiration date prompt ("ENTER EXPIRY DATE" " MMYY" / "DATE D'EXPIRATION" "MMAA"). The card number and the expiration date will be encrypted if VSP keys are loaded and VSP is active on the device.

This prompt will display until the card is removed or, if the TIMEOUT parameter is configured in the XPI configuration file, the terminal times out with the value configured in this parameter. If the timeout is reached, the PIN pad will respond with a TIMEDOUT ('02') response.

Upon successful execution, response can be obtained from property vfiCardData

Card Entry method controlled by the ENTRYMODE XPI configuration parameter on the e210 PINPad:

- ENTRYMODE=0 (Default) Both Manual and Swipe are supported
- ENTRYMODE=1 Manual Entry Only is Supported
- ENTRYMODE=2 Swipe Only is supported

XPI reference: S16

Parameters

lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.

Return values

Result	Code: • 00: OK Key pressed. Retrive card data from vfiCardData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed • 88: No encryption module is activated

10.36.2.123 - (int) obtainCardData: (int) *lcode* displayIdle:(BOOL) *display*

Obtain Card Data (Manual Entry or Swipe)

This command prompts the customer for card entry ("ACCOUNT NUMBER / NO COMPTE") and returns the track data or appropriate response code if the card is swiped. If VSP is enabled, the track data will be encrypted by the VSP module.

If card number is manually entered, the entry will be followed by the expiration date prompt ("ENTER EXPIRY DATE" " MMYY" / "DATE D'EXPIRATION" "MMAA"). The card number and the expiration date will be encrypted if VSP keys are loaded and VSP is active on the device.

This prompt will display until the card is removed or, if the TIMEOUT parameter is configured in the XPI configuration file, the terminal times out with the value configured in this parameter. If the timeout is reached, the PIN pad will respond with a TIMEDOUT ('02') response.

Upon successful execution, response can be obtained from property vfiCardData.

Card Entry method controlled by the ENTRYMODE XPI configuration parameter on the e210 PINPad:

- ENTRYMODE=0 (Default) Both Manual and Swipe are supported
- ENTRYMODE=1 Manual Entry Only is Supported
- ENTRYMODE=2 Swipe Only is supported

XPI reference: S16

Parameters

lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
display	Flag to determine if idle screen is displayed after execution

Return values

Result	Code: • 00: OK Key pressed. Retrive card data from vfiCardData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed • 88: No encryption module is activated

10.36.2.124 - (int) obtainCardData: (int) *lcode* displayIdle:(BOOL) *display* displayExpiry:(BOOL) *expiry* luhnCheck:(BOOL) *luhn* entryMode:(int) *mode*

Obtain Card Data (Manual Entry or Swipe)

This command prompts the customer for card entry ("ACCOUNT NUMBER / NO COMPTE") and returns the track data or appropriate response code if the card is swiped. If VSP is enabled, the track data will be encrypted by the VSP module.

If card number is manually entered, the entry will be followed by the expiration date prompt ("ENTER EXPIRY DATE" " MMYY" / "DATE D'EXPIRATION" "MMAA"). The card number and the expiration date will be encrypted if VSP keys are loaded and VSP is active on the device.

This prompt will display until the card is removed or, if the TIMEOUT parameter is configured in the XPI configuration file, the terminal times out with the value configured in this parameter. If the timeout is reached, the PIN pad will respond with a TIMEDOUT ('02') response.

Upon successful execution, response can be obtained from property vfiCardData.

XPI reference: S16

Parameters

lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
display	Flag to determine if idle screen is displayed after execution
expiry	Flag to determine to display expiration date prompt
luhn	Flag to determine if Luhn check is required
mode	Card Entry Mode: • 0: Use ENTRYMODE config parameter • 1: Manual Only • 2: Swipe Only • 3: Manual and Swipe

Return values

Result	Code: • 00: OK Key pressed. Retrive card data from vfiCardData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed • 88: No encryption module is activated

10.36.2.125 - (int) obtainCardData: (int) *lcode* displayIdle:(BOOL) *display* displayExpiry:(BOOL) *expiry* luhnCheck:(BOOL) *luhn* entryMode:(int) *mode* cvvPrompting:(int) *cvvPrompt* masking:(BOOL) *enableMasking* timeout:(int) *timeout*

Obtain Card Data (Manual Entry or Swipe)

This command prompts the customer for card entry ("ACCOUNT NUMBER / NO COMPTE") and returns the track data or appropriate response code if the card is swiped. If VSP is enabled, the track data will be encrypted by the VSP module.

If card number is manually entered, the entry will be followed by the expiration date prompt ("ENTER EXPIRY DATE" " MMYY" / "DATE D'EXPIRATION" "MMAA"). The card number and the expiration date will be encrypted if VSP keys are loaded and VSP is active on the device.

This prompt will display until the card is removed or, if the TIMEOUT parameter is configured in the XPI configuration file, the terminal times out with the value configured in this parameter. If the timeout is reached, the PIN pad will respond with a TIMEDOUT ('02') response.

Upon successful execution, response can be obtained from property vfiCardData.

XPI reference: S16

Parameters

lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
display	Flag to determine if idle screen is displayed after execution
expiry	Flag to determine to display expiration date prompt
luhn	Flag to determine if Luhn check is required
mode	Card Entry Mode: • 0: Use ENTRYMODE config parameter • 1: Manual Only • 2: Swipe Only • 3: Manual and Swipe

cvvPrompt	Enable CVV prompting: • 0: No prompt (default) • 1: Prompt for CVV and force entry • 2: Prompt but allow to bypass
enableMasking	Flag to determine if delayted masking of entry is enabled
timeout	Timeout value set between 1-999 seconds. 0 defaults to 30 seconds

Return values

Result	Code: • 00: OK Key pressed. Retrive card data from vfiCardData • 01: Unsuccessful • 02: Timed out • 03: CANCEL key was pressed • 04: CORR key was pressed • 88: No encryption module is activated

10.36.2.126 - (int) obtainTrack2Data: (int) *timeout* optionalAmount:(float) *amount*

Obtain Track 2 Data.

This command prompts the customer for card swipe and returns the Track 2 data or appropriate response code.

Upon successful execution, response can be obtained from property vfiCardData.

After the command is executed successfully, the PINPad will display 'PLEASE WAIT', and 'UN MOMENT S.V.P' for 120 seconds. If there is an error, an error message will display for 1 second, and return to idle display for 120 seconds.

- For Tap transactions: The device beeps when the card is tapped indicating the card can be removed. This command only accepts MSD tap.
- Card data might be returned with 01 response code. If the track data does not follow ISO standards, this command will return the 01 error, but with the track data. It will be up to the host application to decide how to handle the track data for non-ISO standard cards.
- If value of 000 is sent for the infinite time-out and Contactless is enabled, the command will time-out at 255 seconds.
- If the response code returned from this message request is 0, then the response data can be retrieved by using the vfiCardData object containing the cardholder entered data value.

XPI reference: S20

Parameters

timeout	A timeout value representing the amount of time, in seconds that the customer has to swipe tap, or insert their card.
amount	Optional amount for the transaction for the prompt display

Return values

Result	Code: • 00: Command was successful. Retrive card data from vfiCardData • 01: Command was unsuccessful or card does not follow ISO standards. If card does not follow ISO standards, any track data will be returned in vfiCardData. • 02: Timed out • 03: Command cancelled • 88: No encryption module is activated

10.36.2.127 - (int) performTGKUpdateRequest

Perform TGK Update Request.

TTGK will be implemented by creating a new command, which will allow for the new PAN and discretionary keys to be retrieved.

Upon successful response, data can be retrieved in vfiCipheredData TKG values

XPI reference: E04

Return values

Result	Code: • 00: Successful. Retrieve TKG values from vfiCipheredData • 01: VSP Module Error, no data. • 02: TGK Error • 99: Error, no VSP / Invalid command format

10.36.2.128 - (int) pinBypass: (BOOL) *bypass*

Set PIN Bypass Support.

This command will enable/disable the PIN bypass for EMV Offline PIN. If enabled, PIN entry can be bypassed using ENTER key. Not all hosts or regions support this feature. The default for this feature is that PIN ByPass is disabled.

Parameters

bypass	TRUE = Enable PIN Entry Bypass, FALSE = Disable PIN Entry bypass

Return values

Result	Code: • 00: Successful - retrieve results in NSArray emaFiles, containing VFIEMAFiles • 01: Unsuccessful.

10.36.2.129 - (int) pinManagementCardAuthorization: (float) *amount* otherAmount:(float) *amtOther* merchantDecision:(int) *decision* minRequestObjects:(NSMutableArray ∗) *reqObjArray*

EMV Transaction Request.

Authorizes the EMV transaction amounts using the inserted chip (EMV) card. This request should follow the response to a request to getCardData:language:amount:otherAmount: if the response indicates the card was inserted.

If result code returns 0-Successful, then the card data can be read from vfiEMVAuthorization

XPI reference: C52

Parameters

amount	Transaction amount value
amtOther	Other transaction amount value
decision	The merchant decision notes additional handling for the EMV request based on required processor handling. Values are: • 0: Request a TC • 1: Force online – 2: Force decline (AAC)
reqObjArray	A dictionary containing EMV tags of type NSData that is required at this transaction stage to be retrieved from the inserted card. See setEMVTags: for an example of creating a dictionary of EMV tags.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData • 22: Chip Error • 23: Card removed • 24: Card Blocked • 25: Card not supported • 26: Card Swiped during Fallback mode • 42: Candidate List Empty

10.36.2.130 - (int) pinManagementCompleteOnlineEMV: (int) *hostDecision* displayResult:(BOOL) *flag* hostTags:(NSMutableDictionary ∗) *hostTagData*

Complete Online EMV Transaction for PIN Management.

This command is required to complete an online EMV transaction. EMV tags returned in vfiEMVCompletionData emvTags variable

XPI reference: C54

Parameters

hostDecision	indicator from the host response; valid values are:: • 0: Host Authorized • 1: Host Declined • 2: Failed to Connect
flag	Indicator to note whether or not to display the final response message; the valid values are: • 0: Do not display • 1: Display
hostTagData	NSMutableDictionary containing EMV tags. The dictionary has the `<NSString>` for keys representing EMV Tag name, and `<NSString>` for objects representing tag data.

Return values

Result	Code: • 00: Success - Manual Entry. Data has been stored in vfiCardData • 01: Invalid command code • 02: Invalid data format • 03: Response has more packs • 04: Previous step missing • 05: Invalid Configuration • 06: Timed out • 07: Timer error • 08: Operation Cancelled • 09: Communication Error • 10: Chip Reader Failure • 21: Success - Swiped Card. Data has been stored in vfiCardData – 22: Chip Error – 23: Card removed – 24: Card Blocked – 25: Card not supported – 26: Card Swiped during Fallback mode – 42: Candidate List Empty

Converting TVL to NSMutableDictionary

EMV data is usually received in TLV (Tag, Length, value) format: `950500000080009B02E8009F2701018A025-`

A339F26080C552B9364D55CE5

This data contains the following EMV tags/values:

Tag	Length	Value
95	05	0000008000
9B	02	E800
9F27	01	01
8A	02	5A33
9F26	08	0C552B9364D55CE5

An example how to create an NSMutableDictionary with these values follows.

```
-(void) setTagsAndSend{

    NSMutableDictionary *emvTags = [[NSMutableDictionary alloc] initWithCapacity:0];

    [emvTags setObject:@"0000008000" forKey:@"95"];
    [emvTags setObject:@"E800" forKey:@"9B"];
    [emvTags setObject:@"01" forKey:@"9F27"];
    [emvTags setObject:@"5A33" forKey:@"8A"];
    [emvTags setObject:@"0C552B9364D55CE5" forKey:@"9F26"];
    [emvTags setObject:@"01" forKey:@"C2"];

    [self pinManagementCompleteOnlineEMV:0 displayResult:0 hostTags:emvTags];

}
```

10.36.2.131 - (void) processReceivedData: (NSData *) *data*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.36.2.132 - (void) PS01: (NSString *) *symbol*

Set Currency Symbol.

See Also

- setCurrencySymbol:

10.36.2.133 - (void) PS02: (int) *timeout*

10.36.2.134 - (void) Q40

Toggle the Card Reader On.

Deprecated Replaced by Q42

See Also

- enableMSR

10.36.2.135 - (void) Q41

Toggle the Card Reader Off.

See Also

- disableMSR

10.36.2.136 - (void) Q42

Toggle the Card Reader On.

See Also

- enableMSRDualTrack

10.36.2.137 - (int) removeCard

Card Remove Request.

Requests the cardholder to remove the card from the PINPad

XPI reference: I02

Return values

Result	Code: • 00: Success - Card Removed • 01: Unsuccessful

10.36.2.138 - (int) requestEMAFiles

Request EMA Files.

This command requests the PIN pad to return all the .EMA files installed on the PIN pad device. Note: If XPI is not installed on GID 1, it can only return the EMA files installed in GID where XPI resides.

Return values

Result	Code: • 00: Successful - retrieve results in NSArray emaFiles, containing VFIEMAFiles • 02: Unsuccessful. Error Occurred

```
int response = [[Vx600 pinPad] requestEMAFiles];
if (response == 0) {
  NSArray *ar = [[Vx600 pinPad] emaFiles];
    for (int x = 0; x < [ar count]; x++) {
      VFIEMAFiles *f = [ar objectAtIndex:x];
       NSLog(@"Rec#%i, GID = %i", x,f.GID);
       NSLog(@"Rec#%i, FileName = %@", x,f.FileName);
       NSLog(@"Rec#%i, Logical = %@", x,f.Logical);
       NSLog(@"Rec#%i, NAME = %@", x,f.NAME);
```

```
        NSLog(@"Rec#%i, VERSION = %@", x,f.VERSION);
        NSLog(@"Rec#%i, ALLOW_MESSAGES = %@", x,f.ALLOW_MESSAGES);
        NSLog(@"Rec#%i, DOWNLOAD_ON_TIMEOUT = %@", x,f.DOWNLOAD_ON_TIMEOUT);
    }
}
```

10.36.2.139 - (int) requestPINRentry: (NSString *) *accountNumber*

Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.

PAN will be sent to PIN pad and PIN block will be generated and sent in response. XPI Reference: S67

See Also

- requestPINRentry:

Return values

Result	Code: • 00: Successful, check for data (if any) in vfiEncryptionData • 01: Unsucessful • 02: Timed Out • 03: CANCEL key was pressed • 04: CORR key was pressed • 06: Invalid Account # Length

10.36.2.140 - (int) requestPINRentry: (NSString *) *accountNumber* pinpadPrompt:(int) *prompt* timeout:(int) *timeout*

Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.

PAN will be sent to PIN pad and PIN block will be generated and sent in response. XPI Reference: S67

See Also

- requestPINRentry:

Parameters

accountNumber	The account number of the card
prompt	value to indicate the prompt to be displayed • 0: Enter PIN • 1: Re-Enter PIN • 2: Enter Current PIN • 3: Enter Assigned / PIN
timeout	Number of seconds (001 - 255) timer should wait for a PIN entry prompt

Return values

Result	Code: • 00: Successful, check for data (if any) in vfiEncryptionData • 01: Unsucessful • 02: Timed Out • 03: CANCEL key was pressed • 04: CORR key was pressed • 06: Invalid Account # Length

10.36.2.141 - (int) requestStringInput: (int) *echoFlag* timeout:(int) *timeout* maxLen:(int) *length*

Request String Input.

This command requests a string of key input from the PIN pad keyboard. It is sent after the Z2 displayMessage:clearScreen: or Z3 displayMessages:message1:message2:clearScreen: command initiating the PIN pad to display one or more messages requesting customer key entries.

After the string of keys (from '0' to '9') is entered, the PIN pad sends a string of ASCII key values corresponding to the key inputs to the copyStringResponse. The PIN pad may echo the input on its display. If no key input is received within the timeout period, the PIN pad will return to its idle state.

XPI reference: Z50.

Parameters

echoFlag	Indicates whether to echo key input with an asterisk '∗', keyed input, or none at all • 0: Echo key input with '∗' • 1: Echo key input • 2: Don't echo key input
timeout	Sets the input timeout in seconds (0-255)
length	Maximum length of the input string allowed (1-49, 49 is the default)

Return values

Result	Code: • 01: Success - Retrive value by calling copyStringResponse • 02: Timeout. • 03: NULL. Enter key was pressed without any other input

10.36.2.142 - (int) requestStringInput: (int) *echoFlag* timeout:(int) *timeout* maxLen:(int) *length* entryType:(int) *type*

Request String Input.

This command requests a string of key input from the PIN pad keyboard. It is sent after the Z2 displayMessage:clearScreen: or Z3 displayMessages:message1:message2:clearScreen: command initiating the PIN pad to display one or more messages requesting customer key entries.

After the string of keys (from '0' to '9') is entered, the PIN pad sends a string of ASCII key values corresponding to the key inputs to the copyStringResponse. The PIN pad may echo the input on its display. If no key input is received within the timeout period, the PIN pad will return to its idle state.

XPI reference: Z50.

Parameters

echoFlag	Indicates whether to echo key input with an asterisk '∗', keyed input, or none at all • 0: Echo key input with '∗' • 1: Echo key input • 2: Don't echo key input
timeout	Sets the input timeout in seconds (0-255)
length	Maximum length of the input string allowed (1-49, 49 is the default)
type	Entry type. Valid values are: • 0= Alphanumeric (Default) • 1= Numeric

Return values

Result	Code: • 00: Success - Retrive value by calling copyStringResponse • 01: No Response. Enter key was pressed without any other input • 02: Timeout

10.36.2.143 - (void) restartLoopDelay: (float) *sec*

Controls the restart loop delay.

Parameters

sec	The amount of time in fractional sections to wait between attempts in establishing contact to e210/e255/e315 while waiting for initialization

Default is 1.0 seconds.

10.36.2.144 - (int) retrieveEParmsData

Retrieve EParms Data.

The request for EParms data - this method will be used in conjunction with the card data retrieval commands – 81, C30/31, S16, and S20. The request for EParms data (E02) should be sent to XPI once the entered card data is returned by the command. The information will be retrieved from the VCL, and returned to the controller in the EParms response.

Upon successful response, data can be retrieved in vfiCipheredData EParms

XPI reference: E02

Return values

Result	Code: • 00: Successful. Retrieve EParms from vfiCipheredData • 01: VSP Module Error, no data. • 02: EParms Error • 99: Error, no VSP / Invalid command format

10.36.2.145 - (void) S00

Cancel Command.

See Also

- cancelCommand

10.36.2.146 - (void) S01

Query and Initialize Idle Prompt.

See Also

- initializeIdle

10.36.2.147 - (void) S07: (int) *displayTimer* pin:(int) *pinTimer* balance:(int) *balanceTimer*

Store Timer Values.

See Also

- storeTimers:pin:balance:

10.36.2.148 - (int) S11: (NSString ∗) *msg1* line2:(NSString ∗) *msg2*

Display Message and Wait for Key.

See Also

- displayMessageAndWait:line2:

10.36.2.149 - (int) S11: (NSString ∗) *msg1* line2:(NSString ∗) *msg2* line3:(NSString ∗) *msg3* line4:(NSString ∗) *msg4*

Display Message and Wait for Key.

See Also

- displayMessageAndWait:line2:line3:line4:

10.36.2.150 - (int) S12: (NSString ∗) *msgID1* message2:(NSString ∗) *msgID2* minLength:(int) *minLen* maxLength:(int) *maxLen* language:(int) *lcode* allowSkip:(BOOL) *flag* secureEntry:(BOOL) *secure*

General Prompt Entry.

See Also

- generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry:

10.36.2.151 - (int) S12: (NSString ∗) *msgID1* message2:(NSString ∗) *msgID2* minLength:(int) *minLen* maxLength:(int) *maxLen* language:(int) *lcode* allowSkip:(BOOL) *flag* secureEntry:(BOOL) *secure* displayCanceled:(BOOL) *display* restoreScreen:(BOOL) *restore*

General Prompt Entry.

See Also

- generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry:displayCanceled:restoreScreen:

10.36.2.152 - (int) S12Default: (NSString ∗) *msgID1* message2:(NSString ∗) *msgID2* minLength:(int) *minLen* maxLength:(int) *maxLen* language:(int) *lcode* initialValue:(NSString ∗) *defaultValue* allowSkip:(BOOL) *flag*

General Prompt Entry.

See Also

- generalPromptDefault:message2:minLength:maxLength:language:initialValue:allowSkip:

10.36.2.153 - (int) S13: (NSString ∗) *msg1* message2:(NSString ∗) *msg2* choice1:(NSString ∗) *cho1* choice2:(NSString ∗) *cho2* choice3:(NSString ∗) *cho3* choice4:(NSString ∗) *cho4*

Display Menu.

See Also

- menuChoiceSelection:message2:choice1:choice2:choice3:choice4:

10.36.2.154 - (void) S14: (NSString ∗) *line1* Line2:(NSString ∗) *line2* Line3:(NSString ∗) *line3* Line4:(NSString ∗) *line4*

Display Messages.

See Also

- displayMessages:Line2:Line3:Line4:

10.36.2.155 - (void) S14: (NSString ∗) *line1* Line2:(NSString ∗) *line2* Line3:(NSString ∗) *line3* Line4:(NSString ∗) *line4* timeout:(int) *timeout*

Display Messages.

See Also

- displayMessages:Line2:Line3:Line4:timeout:

10.36.2.156 - (int) S16: (int) *lcode*

Obtain Card Data (Manual Entry or Swipe)

See Also

- obtainCardData:

10.36.2.157 - (int) S16: (int) *lcode* displayIdle:(BOOL) *display*

Obtain Card Data (Manual Entry or Swipe)

See Also

- obtainCardData:displayIdle:

10.36.2.158 - (int) S16: (int) *lcode* displayIdle:(BOOL) *display* displayExpiry:(BOOL) *expiry* luhnCheck:(BOOL) *luhn* entryMode:(int) *mode*

Obtain Card Data (Manual Entry or Swipe)

See Also

- obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:

10.36.2.159 - (int) S16: (int) *lcode* displayIdle:(BOOL) *display* displayExpiry:(BOOL) *expiry* luhnCheck:(BOOL) *luhn* entryMode:(int) *mode* cvvPrompting:(int) *cvvPrompt* masking:(BOOL) *enableMasking* timeout:(int) *timeout*

Obtain Card Data (Manual Entry or Swipe)

See Also

obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout()

10.36.2.160 - (int) S20: (int) *timeout* optionalAmount:(float) *amount*

Obtain Track 2 Data.

See Also

- obtainTrack2Data:optionalAmount:

10.36.2.161 - (int) S21: (float) *amount* language:(int) *lcode* tip:(BOOL) *tipFlag* cashback:(BOOL) *cashbackFlag* surcharge:(float) *surchgAmt*

Data Entry Request.

See Also

- dataEntryRequest:language:tip:cashback:surcharge:

10.36.2.162 - (int) S21prompt: (NSString ∗) *prompt*

Data Entry Request.

See Also

- dataEntryRequestPrompt:

10.36.2.163 - (int) S23: (float) *amount* language:(int) *lcode* transID:(int) *ID* menuArray:(NSString ∗) *promptArray* surcharge:(float) *surchgAmt* accountNum:(NSString ∗) *account* tipOptions:(NSString ∗) *tipOpt* cashBackOptions:(NSString ∗) *cashBackOpt* surchargeOptions:(int) *surchargeOpt*

Combined Prompt Command.

See Also

- combinedPromptCommand:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBackOptions-:surchargeOptions:

10.36.2.164 - (int) S66: (NSData ∗) *macData*

Create Mac Request.

See Also

- createMac:

10.36.2.165 - (int) S67: (NSString ∗) *accountNumber*

Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.

PAN will be sent to PIN pad and PIN block will be generated and sent in response. XPI Reference: S67

See Also

- requestPINRentry:

Return values

Result	Code: • 00: Successful, check for data (if any) in vfiEncryptionData • 01: Unsucessful • 02: Timed Out • 03: CANCEL key was pressed • 04: CORR key was pressed • 06: Invalid Account # Length

10.36.2.166 - (int) S67: (NSString *) *accountNumber* pinpadPrompt:(int) *prompt* timeout:(int) *timeout*

Request PIN Re-entry This command re-prompts the 'Enter PIN' prompt.

PAN will be sent to PIN pad and PIN block will be generated and sent in response. XPI Reference: S67

See Also

- requestPINRentry:

Parameters

accountNumber	The account number of the card
prompt	value to indicate the prompt to be displayed • 0: Enter PIN • 1: Re-Enter PIN • 2: Enter Current PIN • 3: Enter Assigned / PIN
timeout	Number of seconds (001 - 255) timer should wait for a PIN entry prompt

Return values

Result	Code: • 00: Successful, check for data (if any) in vfiEncryptionData • 01: Unsucessful • 02: Timed Out • 03: CANCEL key was pressed • 04: CORR key was pressed • 06: Invalid Account # Length

10.36.2.167 - (int) S68

Test PIN Pad.

See Also

- testPINPad

10.36.2.168 - (int) S70: (int) *transId* language:(int) *lcode* amount:(float) *amt* pan:(NSString ∗) *acctNumber* mac:(NSData ∗) *macData*

Perform Interac Request Transaction.

See Also

- doInteracRequest:language:amount:pan:mac:

10.36.2.169 - (int) S71: (NSData ∗) *macBlock* language:(int) *lcode* pinKey:(NSData ∗) *tpkKey* macKey:(NSData ∗) *takKey* msgLine1:(NSString ∗) *msg1* msgLine2:(NSString ∗) *msg2* macData:(NSData ∗) *data*

Perform Interac Response Analysis.

See Also

- validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData:

10.36.2.170 - (int) S71Fields: (NSData ∗) *macBlock* language:(int) *lcode* msgLine1:(NSString ∗) *msg1* msgLine2:(NSString ∗) *msg2* macData:(NSData ∗) *macdata* identity1:(NSData ∗) *identity1* data1:(NSData ∗) *data1* identityK:(NSData ∗) *identityK* dataK:(NSData ∗) *dataK*

Perform Interac Response Analysis.

See Also

- validateMacResponseFields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK:

10.36.2.171 - (int) S95

Polls e210/e255/e315 for Diagnostic Info.

See Also

- diagnosticInfo

10.36.2.172 - (int) S95Extended

Polls e210/e255/e315 for Diagnostic Info.

See Also

- diagnosticInfoExtended

10.36.2.173 - (int) S97

Request EMA Files.

See Also

- requestEMAFiles

10.36.2.174 - (EncryptionMode) selectEncryptionMode: (EncryptionMode) *mode*

Select Encryption Mode.

This command is used to select the encryption mode to be used during runtime. If PKI is set, VSP will be disabled to prevent double encryption during card swipe. If VSP is set, VSP will be enabled, and PKI encryption will not be used.

The config.sys variable XPIDEFSEC allows the user to set the default encryption mode during startup. If a new value is sent with the E10 command, the parameter value will be updated.

XPI reference: E10

Parameters

mode	EncryptionMode() to set

Return values

EncryptionMode().	If unsuccessful, then returns value of previous setting.

10.36.2.175 - (EncryptionMode) selectEncryptionMode: (EncryptionMode) *mode* keyIndex:(int) *index*

Select Encryption Mode.

This command is used to select the encryption mode to be used during runtime. If PKI is set, VSP will be disabled to prevent double encryption during card swipe.

This command will support VSD mode of encryption. VSD supports 10 DUPKT key slots, indexed from 0 to 9. Optionally the key slot or key index may be specified with E10. Key index defaults to 00 if no index is specified.

The config.sys variable XPIDEFSEC allows the user to set the default encryption mode during startup. If a new value is sent with the E10 command, the parameter value will be updated.

XPI reference: E10

Parameters

mode	EncryptionMode() to set
index	Index of DUKPT key. This optional value is in the range of 0 to 9 and only applicable if Encryption Mode = EncryptionMode_VSD. The default value is 0;

Return values

EncryptionMode().	If unsuccessful, then returns value of previous setting. If return value is EncryptionMode_VSD, then property vsdKeyIndex can be polled for the DUPKT key slot number.

10.36.2.176 - (int) sendDataCommand: (NSMutableData *) *cmd* calcLRC:(BOOL) *lrc*

Send a NSData object to XPI application.

Sends a command represented by the provide NSData object to the XPI application through the accessory protocol.

Parameters

cmd	NSData representation of command to execute
lrc	If `TRUE`, this will wrap command with start/end/lrc: `<STX>cmd<ETX><LRC>`

10.36.2.177 - (int) sendStringCommand: (NSString *) *cmd* calcLRC:(BOOL) *lrc*

Send a string command to XPI application.

Sends a command represented by the provide string object to the XPI application through the accessory protocol.

Parameters

cmd	NSString representation of command to execute
lrc	If `TRUE`, this will wrap command with start/end/lrc: `<STX>cmd<ETX><LRC>`

10.36.2.178 - (void) setAccountEntryTimeout: (int) *seconds*

Time to wait for response to account number request.

When manual PAN entry is requested, a timeout error will be returned from enumeration VFIErrorCode if wait time exceeds Account Entry timeout value. Default value is 45 seconds.

Parameters

seconds	Amount of time to wait for account entry entry in response to manual PAN entry request before timeout

10.36.2.179 - (void) setACKTimeout: (int) *seconds*

Time to wait for `<ACK>` in response to API calls sent to e210/e255/e315.

All commands sent to e210/e255/e315 should return `<ACK>`, followed by data specific for that command (if any). The default time waiting for e210/e255/e315 to `<ACK>` is 2 seconds. If no `<ACK>` is received within the allotted time frame, VMF will attemp to resend the command up to 3 times before error condition generated.

Parameters

seconds	Amount of time to wait for `<ACK>` after sending command

10.36.2.180 - (int) setAdditionalEVMFlags: (BOOL) *visaDebitFlag* ctlsMode:(int) *mode* pdolSupport:(BOOL) *pdolFlag*

Additional EMV Info.

Sets additional EMV Configuration flags stored in the external PINPad.

The default contactless mode is MSD contactless. If the CTLS mode is changed, the device needs to be rebooted to change the configuration file in the CTLS reader. The PINPad application will display 'RESTART REQUIRED \ PRESS OK', upon pressing OK Key, application will reboot. In order to enable the Visa DEBIT flag the visaDebitFlag

should be set to Boolean 'YES' and the modifyCandidateListFlag in the MVT table should be set to 'YES' using the updateEMVConfigRecord:tableData: command task.

XPI reference: D25.

Parameters

visaDebitFlag	Indicates whether VISA Debit Opt-Out is enabled or not
mode	Contactless mode, 0 – EMV Contactless, 1 – MSD Contactless, 2 – CTLS Disabled
pdolFlag	Indicates whether PDOL is supported or not

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.181 - (int) setAdditionalEVMFlags: (BOOL) *visaDebitFlag* ctlsMode:(int) *mode* pdolSupport:(BOOL) *pdolFlag* valueLinkCtls:(BOOL) *valuLinkFlag* amexMode:(int) *amexMode*

Additional EMV Info.

Sets additional EMV Configuration flags stored in the external PINPad.

The default contactless mode is MSD contactless. If the CTLS mode is changed, the device needs to be rebooted to change the configuration file in the CTLS reader. The PINPad application will display 'RESTART REQUIRED \ PRESS OK', upon pressing OK Key, application will reboot. In order to enable the Visa DEBIT flag the visaDebitFlag should be set to Boolean 'YES' and the modifyCandidateListFlag in the MVT table should be set to 'YES' using the updateEMVConfigRecord:tableData: command task.

XPI reference: D25.

Parameters

visaDebitFlag	Indicates whether VISA Debit Opt-Out is enabled or not
mode	Contactless mode, 0 – EMV Contactless, 1 – MSD Contactless, 2 – CTLS Disabled
pdolFlag	Indicates whether PDOL is supported or not. 1=ON, 0=OFF;
valuLinkFlag	Indicates whether Valulink is supported or not. 1=ON, 0=OFF;

amexMode	AMEX Contactless mode, 0 – Default, same as Contactless Mode, 1 – MSD Mode, 2 – EMV Mode

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.182 - (int) setC30CustomPrompts: (NSString ∗) *line1* line2:(NSString ∗) *line2* line3:(NSString ∗) *line3* line4:(NSString ∗) *line4* amt1:(BOOL) *amt1* amt2:(BOOL) *amt2* amt3:(BOOL) *amt3* amt4:(BOOL) *amt4*

Prompt Message Changes.

This command is used to change the messages that are displayed during the C30 command. You can change the message, or leave the message on any line(s) blank and instead have it display the entered amount by setting the amount flag to TRUE

Parameters

line1	Message to display on line 1, up to 16 characters.
line2	Message to display on line 2, up to 16 characters.
line3	Message to display on line 3, up to 16 characters.
line4	Message to display on line 4, up to 16 characters.
amt1	Set to TRUE to display the amount on line 1.
amt2	Set to TRUE to display the amount on line 2.
amt3	Set to TRUE to display the amount on line 3.
amt4	Set to TRUE to display the amount on line 4.

XPI reference: D30

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format • 04: Error Opening File • 05: Error Reading File • 06: Error Writing to File • 07: Error Closing File • 08: Error in Table Record (Future Use) • 09: Table Filename Duplication (Future Use)

10.36.2.183 - (int) setC30DefaultPrompts

Prompt Message Changes.

This command is used to reset the messages that are displayed during the C30 command back to defaults

XPI reference: D30

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format • 04: Error Opening File • 05: Error Reading File • 06: Error Writing to File • 07: Error Closing File • 08: Error in Table Record (Future Use) • 09: Table Filename Duplication (Future Use)

10.36.2.184 - (void) setContinueTransactionTimeout: (int) *seconds*

Time to wait for response during EMV transaction.

When cardAuthorization() method is executed during an EMV transaciton flow, a timeout error will be returned from enumeration VFIErrorCode if wait time exceeds Continue Transaction timeout value. Default value is 45 seconds.

Parameters

seconds	Amount of time to wait for account entry entry in response to manual PAN entry request before timeout

10.36.2.185 - (void) setCurrencySymbol: (NSString ∗) *symbol*

Set Currency Symbol.

This Property Set command will automatically configure the Config parameter XPICURR. XPI reference: PS01

Parameters

symbol	Currency Symbol(s). If empty, no currency symbol will be used in all XPI amount display. Maximum number of characters is 3 characters.

10.36.2.186 - (int) setEMVTag: (NSString ∗) *tag* value:(NSData ∗) *tagValue*

Set EMV Tag Information.

This command is used to set tag information from the inserted card. The request can set one tag.

XPI reference: C19

Parameters

tag	4 character alphanumeric EMV tag in hexadecimal format
tagValue	Value of the EMV tag data to be set for the inserted card

Return values

Result	Code: • 00: Successful • 01: Unsuccessful

10.36.2.187 - (int) setEMVTags: (NSMutableDictionary ∗) *tagArray*

Update EMV Tag Information.

This command is used to set tag information from the inserted card. The request can set multiple tags.

XPI reference: C21

Parameters

tagArray	A dictionary containing NSData EMV tags that will be set on the inserted card

Return values

Result	Code: • 00: Successful • 01: Unsuccessful

Converting TVL to NSMutableDictionary

EMV data is usually received in TLV (Tag, Length, value) format: `950500000080009B02E8009F2701018A025-A339F26080C552B9364D55CE5`

This data contains the following EMV tags/values:

Tag	Length	Value
95	05	0000008000
9B	02	E800
9F27	01	01
8A	02	5A33
9F26	08	0C552B9364D55CE5

An example how to create an NSMutableDictionary with these values follows.

```
-(void) setTagsAndSend{

    NSMutableDictionary *emvTags = [[NSMutableDictionary alloc] initWithCapacity:0];

    [emvTags setObject:@"0000008000" forKey:@"95"];
    [emvTags setObject:@"E800" forKey:@"9B"];
    [emvTags setObject:@"01" forKey:@"9F27"];
    [emvTags setObject:@"5A33" forKey:@"8A"];
    [emvTags setObject:@"0C552B9364D55CE5" forKey:@"9F26"];
    [emvTags setObject:@"01" forKey:@"C2"];

    [self setEMVTags:emvTags];

}
```

10.36.2.188 - (void) setFrameworkTimeout: (int) *seconds*

Time to wait for response from API calls sent to e210/e255/e315.

All commands sent to e210/e255/e315 should return <ACK>, followed by data specific for that command (if any). The default framework timeout waiting for data is 5 seconds. If no data is received, a framework timeout error will be returned from enumeration VFIErrorCode

Parameters

seconds	Amount of time to wait for response before timeout error

10.36.2.189 - (void) setInitLoop: (int) *loop*

Controls the amount of looping attempts to contact XPI application.

Parameters

loop	The number of loops in establishing contact to e210/e255/e315 while waiting for initialization

If the e210/e255/e315 is unresponsive, the framework will loop the specified number of times, with a delay of restartLoopDelay: between each attempt. The default is 59 loops.

10.36.2.190 - (void) setIOTimer: (int) *timeout*

10.36.2.191 - (void) setKSN20Char: (BOOL) *is20Char*

Forces Key Serial Number to 20 characters.

Determines of KSN should always return 20 characters, or just value provided by e210/e255/e315

Parameters

is20Char	Valid values: • `TRUE` Left pad KSN with 0xFF to return value taht is 20 characters • `FALSE` Return KSN unmodified - do not left pad (default behavior).

10.36.2.192 - (void) setParameter: (NSString *) *parmName* value:(NSString *) *parmValue*

Sends a single parameters to the e210/e255/e315.

This will update the config.sys with a single parameter. A device reboot will be required to complete process.

Parameters

parmName	The parameter name
parmValue	The parameter value

```
[pinPad setParameter:@"*APPOVERRIDE" value:@"0"];
```

10.36.2.193 - (void) setParameterArray: (NSMutableArray *) *parameters*

Sends a list of parameters to the e210/e255/e315.

This will update the config.sys with supplied parameters. The parameters are provided as a NSMutableArray with each line being a separate parameter. A device reboot will be required to complete process.

Parameters

parameters	A NSMutableArray containg objects of type NSString

```
NSMutableArray* arr = [[NSMutableArray alloc] initWithCapacity:0];
[arr addObject:@"*APPOVERRIDE=0"];
[arr addObject:@"*STAYCONNECTED=1"];
[arr addObject:@"*KEEPALIVE=1"];
[arr addObject:@"*POWERSHARE=0"];
[arr addObject:@"*CHARGEHOST=50"];
[arr addObject:@"*XCDEBUG="];
[pinPad setParameterArray: arr];
```

10.36.2.194 - (void) setPINTimeout: (int) *seconds*

Time to wait for response to PIN entry.

When commands are executing requiring PIN code entry, a timeout error will be returned from enumeration VFIErrorCode if wait time exceeds PIN timeout value. Default value is 45 seconds.

Parameters

seconds	Amount of time to wait for PIN entry in response to PIN request before timeout

10.36.2.195 - (void) setPromptTimeout: (int) *seconds*

Time to wait for response to prompt.

When commands are executing requiring pinpad input before proceeding, a timeout error will be returned from enumeration VFIErrorCode if wait time exceeds prompt timeout value. Default value is 30 seconds.

Parameters

seconds	Amount of time to wait for pinpad keypress in response to prompt before timeout

10.36.2.196 - (int) setROSParams: (uint) *floorLimit* threshold:(uint) *thresh* target:(uint) *targ* targetPercent:(uint) *percent*

Modify TRM Parameters.

This function can be used to change the value of Terminal Risk Management parameters dynamically. The modified values are used only for the next C32 cardAuthorization:otherAmount:merchantDecision:minRequestObjects: transaction. The values of the parameters will be reverted to their original values from the MVT for the next C32 transaction. It is advisable to issue that command prior to C32 Command

XPI reference: C18

Parameters

floorLimit	New floor limit setting for the subsequent card authorization transaction
thresh	New RS Threshold amount for the subsequent card authorization transaction
targ	New RS Target percentage setting for the subsequent card authorization transaction
percent	New RS Max Percent setting for the subsequent card authorization transaction

Return values

Result	Code: • 00: Successful • 01: Unsuccessful

10.36.2.197 + (VFIPinpad ∗) sharedController

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.36.2.198 - (void) startServer

Start the TCP server to listen for connections.

10.36.2.199 - (int) statusCTLS

Retrieve Status of Contactless Reader.

This command is used to retrieve the current status of Contactless Reader and OPTFLAG. XPI reference: D27. If successful in retrieving the Contactless State, then the reader state can be retrieved by calling ctlsState and the OPTFlag Values can be retreived by calling ctlsOPTFLAG.

Return values

Result	Code: • 00: Successful in retrieving the Contactless Reader State - retrieve values by calling ctlsState and ctlsOPTFLAG • 01: Contactless is being initialized. If this command is issued immediately after initializeContactless:sig:CVM: or setAdditionalEVMFlags:ctlsMode:pdolSupport:. Contactless Reader State returned will be 00. • 02: Failure. Possible cause is VxCI is not installed • 03: Failure. Contactless is disabled via OPTFLAG.

10.36.2.200 - (void) stopServer

Stops the TCP server from listening for connections. Terminates any current connection.

10.36.2.201 - (void) storeTimers: (int) *displayTimer* pin:(int) *pinTimer* balance:(int) *balanceTimer*

Store Timer Values.

This command will store the Prompt Display, PIN Inter-digit, and Display Balance timers.

Passing a value of 0 to any of the timers will not alter their values. Only a value of greater than zero (0) will change the timer values. The default value for the Prompt Timer and the Inter-digit PIN Timer is 15 seconds. The default value for the Balance Display Timer is 5 seconds.

XPI reference: S07

Parameters

displayTimer	Prompt Display Timer
pinTimer	Inter-digit PIN Entry Timer (time-out value between the entry of each digit of the PIN during the pin entry request)
balanceTimer	Account Balance Display Timer

10.36.2.202 - (int) testPINPad

Test PIN Pad.

This command tests the working keys in the PIN pad. Prompt for PIN will be displayed - any PIN can be entered from 4-12 digits. PIN Block will be calculated with Account Number '000000000000' (12-'0's). Result can be obtained in vfiEncryptionData pinBlock and pinBlockStr. XPI reference: S68.

Return values

Result	Code: • 00: Successful. Evaluate vfiEncryptionData pinBlock/pinBlockStr for PIN Block • 01: Unsuccessful, no data

10.36.2.203 - (int) updateAIDRecord: (int) *recno* AID:(int) *aidNo* partialNameMatch:(BOOL) *flag* supportedAID:(NSData ∗) *supAID* termAVN:(NSData ∗) *avn* term2ndAVN:(NSData ∗) *avn2* recommendedAIDName:(NSString ∗) *aidName*

Update AID in EST (EMV Card Scheme Table)

Updates the AID in the EMV Scheme Table (EST) stored in the external PINPad.

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D13.

Parameters

recno	Record number to update (e.g. 01, 02, 03 . . .)
aidNo	AID number to update (e.g. 01, 02, 03 . . .)
flag	Boolean indicator where 'true' is partial matching is allowed, and 'false' is partial matching is not allowed.
supAID	32 bytes, including NULL; Ex. A0000000031010, A0000000031020
avn	Terminal AVN, 6 bytes
avn2	2nd Terminal AVN, 6 bytes
aidName	Recommended AID Name, 16 bytes (e.g. 'VISA')

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.204 - (int) updateEMVConfigRecord: (int) *recno* tableData:(VFIEMVConfiguration ∗) *emvData*

Update MVT (EMV Configuration Table)

Updates the EMV Configuration Table (MVT) stored in the external PINPad.

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D14.

Parameters

recno	Record # to be updated with the provided data
emvData	Configuration data contained within the VFIEMVConfiguration data object

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.205 - (int) updateExtendedEMVRecord: (int) *recno* mvtIndex:(int) *emvIndex* groupName:(NSString ∗) *group* ctlsFloorLimit:(float) *ctlsFloor* ctlsCVMLimit:(float) *ctlsCVM* ctlsTranLimit:(float) *ctlsTran* ctlsTacDefault:(NSData ∗) *tacDefault* ctlsTacDenial:(NSData ∗) *tacDenial* ctlsTacOnline:(NSData ∗) *tacOnline* visaTTQ:(NSData ∗) *ttq*

Update EMVT.

This command will update the EMVT table with the EMV Contactless settings for the reader to make decisions during the transaction per card scheme. The command should be sent once for each record number (once for each card scheme record).

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D19.

Parameters

recno	Represents the record number to be updated (e.g. 000, 001, 002 . . .)
emvIndex	EMV Configuration Table Index (00-99); should be the same as the EMV index in the EEST table for the same card type
group	Identifying group name, 20 bytes right padded with 0x00 (e.g. “VISA”, “MASTERCARD” . . .)
ctlsFloor	Contactless floor limit; can be up to a 12 digit value including a 2-digit decimal
ctlsCVM	Contactless CVM limit; can be up to a 12 digit value including a 2-digit decimal
ctlsTran	Contactless transaction limit; can be up to a 12 digit value including a 2-digit decimal
tacDefault	TAC Default value, 10-byte hexadecimal

tacDenial	TAC Denial value, 10-byte hexadecimal
tacOnline	TAC Online value, 10-byte hexadecimal
ttq	Visa TTQ, 8-byte hexadecimal

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.206 - (int) updateExtendedEMVRecord: (int) *recno* mvtIndex:(int) *emvIndex* groupName:(NSString ∗) *group* ctlsFloorLimit:(float) *ctlsFloor* ctlsCVMLimit:(float) *ctlsCVM* ctlsTranLimit:(float) *ctlsTran* ctlsTacDefault:(NSData ∗) *tacDefault* ctlsTacDenial:(NSData ∗) *tacDenial* ctlsTacOnline:(NSData ∗) *tacOnline* visaTTQ:(NSData ∗) *ttq* termCapabilities:(NSString ∗) *termCapb* addTermCapabilities:(NSString ∗) *addTermCapb*

Update EMVT.

This command will update the EMVT table with the EMV Contactless settings for the reader to make decisions during the transaction per card scheme. The command should be sent once for each record number (once for each card scheme record).

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D19.

Parameters

recno	Represents the record number to be updated (e.g. 000, 001, 002 . . .)
emvIndex	EMV Configuration Table Index (00-99); should be the same as the EMV index in the EEST table for the same card type
group	Identifying group name, 20 bytes right padded with 0x00 (e.g. “VISA”, “MASTERCARD” . . .)
ctlsFloor	Contactless floor limit; can be up to a 12 digit value including a 2-digit decimal
ctlsCVM	Contactless CVM limit; can be up to a 12 digit value including a 2-digit decimal
ctlsTran	Contactless transaction limit; can be up to a 12 digit value including a 2-digit decimal
tacDefault	TAC Default value, 10-byte hexadecimal

tacDenial	TAC Denial value, 10-byte hexadecimal
tacOnline	TAC Online value, 10-byte hexadecimal
ttq	Visa TTQ, 8-byte hexadecimal
termCapb	Represents the card data input, CVM, and security capabilities of the device.
addTermCapb	Represents the data input and output capabilities of the device.

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.207 - (int) updateExtendedEMVRecord: (int) *recno* mvtIndex:(int) *emvIndex* groupName:(NSString ∗) *group* ctlsFloorLimit:(float) *ctlsFloor* ctlsCVMLimit:(float) *ctlsCVM* ctlsTranLimit:(float) *ctlsTran* ctlsTacDefault:(NSData ∗) *tacDefault* ctlsTacDenial:(NSData ∗) *tacDenial* ctlsTacOnline:(NSData ∗) *tacOnline* visaTTQ:(NSData ∗) *ttq* termCapabilities:(NSString ∗) *termCapb* addTermCapabilities:(NSString ∗) *addTermCapb* ctlsCountry:(NSString ∗) *country* ctlsCurrency:(NSString ∗) *currency*

Update EMVT.

This command will update the EMVT table with the EMV Contactless settings for the reader to make decisions during the transaction per card scheme. The command should be sent once for each record number (once for each card scheme record).

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D19.

Parameters

recno	Represents the record number to be updated (e.g. 000, 001, 002 . . .)
emvIndex	EMV Configuration Table Index (00-99); should be the same as the EMV index in the EEST table for the same card type
group	Identifying group name, 20 bytes right padded with 0x00 (e.g. “VISA”, “MASTERCARD” . . .)
ctlsFloor	Contactless floor limit; can be up to a 12 digit value including a 2-digit decimal

ctlsCVM	Contactless CVM limit; can be up to a 12 digit value including a 2-digit decimal
ctlsTran	Contactless transaction limit; can be up to a 12 digit value including a 2-digit decimal
tacDefault	TAC Default value, 10-byte hexadecimal
tacDenial	TAC Denial value, 10-byte hexadecimal
tacOnline	TAC Online value, 10-byte hexadecimal
ttq	Visa TTQ, 8-byte hexadecimal
termCapb	Represents the card data input, CVM, and security capabilities of the device.
addTermCapb	Represents the data input and output capabilities of the device.
country	Country code used for contactless transactions • 840 = US • 124 = CANADA
currency	Currency code used for contactless transactions • 840 = US • 124 = CANADA

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.208 - (int) updateExtendedSchemeRecord: (int) *recno* emvRecord:(int) *emvRecno* AID:(NSString ∗) *aid* maxAIDLen:(int) *maxLen* appFlow:(int) *flow* partialName:(BOOL) *pflag* acctSelect:(BOOL) *aflag* enableCtls:(BOOL) *cflag* transScheme:(int) *scheme*

Update EEST Table.

Updates the Extended EMV Scheme Table (EEST) stored in the external PINPad.

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D17.

Parameters

recno	Represents the record number to be updated (e.g. 000, 001, 002 . . .)
emvRecno	EMV Record Number to be updated (00-99)
aid	Supported AID, 32 bytes padded with 0x00
maxLen	Maximum AID Length
flow	'01' <MASTERCARD>, '02' AMEX, '03' MC MSTRIPE, '06' <VISA>, '13' DISCOVER, '14' JCB, '21' INTERAC
pflag	Partial Name flag
aflag	Account Selection flag
cflag	Contactless enable flag
scheme	Decimal representation of Transaction Scheme bit field • MasterCard AIDs : 129 – MSD Only, 193 – MSD and MChip • Visa AIDs : 128 – MSD Only, 192 – MSD and qVSDC • Interac : 128 – MSD Only • American Express AID : 194 – MSD Only • Discover AID : 128 – MSD Only

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.209 - (void) updateFromUrl: (NSString *) *theURL*

Updates e210/e255/e315 from file specified by URL.

Updates the e210/e255/e315 with an properly created archive located on a network share, either intranet or internet.

Parameters

theURL	Network location specifying the valid .zip archive for upload into e210/e255/e315

10.36.2.210 - (void) updateFromZip: (NSData ∗) *data*

Updates e210/e255/e315 from a .zip file.

Updates the e210/e255/e315 with an properly created archive compressed as a .zip archive.

Parameters

data	The .zip archive containing data to transfer into e210/e255/e315

```
NSFileManager *fm;
fm = [NSFileManager defaultManager];
NSString *documentsDirectory = [[NSBundle mainBundle] resourcePath];
NSString* currentFile = [[NSString alloc] initWithFormat:@"%@/%@",documentsDirectory,@"
      XPI530_OSApps_Gen25.zip"];
NSData* dt = [[NSData alloc] initWithContentsOfFile:currentFile];
[pinPad updateFromZip:dt];
```

10.36.2.211 - (int) updateICCT: (int) *recno* ctlsMerchantType:(int) *merchType* ctlsTermTransInfo:(NSString ∗) *transInfo* ctlsTermTransType:(NSString ∗) *transType* ctlsReqReceiptLimit:(float) *ctlsReceipt* ctlsOptionStatus:(NSString ∗) *optionStatus* ctlsReaderFlooorLimit:(float) *ctlsFloor*

Update ICCT.

This command will update the Interac Contactless Configuration (ICCT) Table with the parameters required in order to configure the CTLS reader to accept Interac cards.

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D20.

Parameters

recno	Record number of the ICCT (e.g. 000, 001, 002 . . .)
merchType	Merchant Type Indicator used by the card to choose limits for card risk management. Valid values 01 – 05
transInfo	Provides Terminal Transaction Information such as; display capability, Interac contact interface application support, online capabilities, and card read capabilities (mag stripe, contact). Tag 9F59
transType	A dynamic value that must to updated to provide the reader with the current transaction type. 9F5A
ctlsReceipt	Terminal Contactless Receipt Required Limit – Limit amount used to compare against Transaction amount to automatically print a transaction record. Right padded with '0', include 2 digit decimal. 9F5D
optionStatus	Options supported by the terminal defining how to process transactions using a different currency or country code. 9F5E
ctlsFloor	Floor limit amount used to compare against Transaction amount. 9F5F

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.212 - (int) updatePassThru: (int) *recordNum* aid:(NSString ∗) *aid* version:(NSString ∗) *version*

Update PassThru Configuration File.

This command will update a record used in PassThru mode to configure the reader to accept the ValueLink card.

XPI reference: D26.

Parameters

recordNum	Record number 01 - 10
aid	AID 5 to 34 characters
version	Application Version Name (ASCII Hex string)

Return values

Result	Code: • 00: Successful • 02: Invalid Parameter • 03: Failure on updating (Write Failed)

10.36.2.213 - (int) updateSchemeRecord: (int) *recno* numberTrans:(long) *numTrans* scheme:(NSString ∗) *schemeLabel* regId:(NSString ∗) *RID* CSNList:(NSString ∗) *csn* emvTableRecord:(long) *rec* CSNFile:(NSString ∗) *csnFile*

Update Scheme Record.

Updates Registered Application Provider ID (RID) information in the EMV Scheme Table (EST) stored in the external PINPad. CAPK files will be removed if they are named in the scheme record when the new record is updated.

Knowledge of the EMV Scheme Table (EST) is required in order to properly manage updates. Refer to the EST Record Structure in the API Specification for XPI.

XPI reference: D15.

Parameters

recno	Represents the record number to be updated (e.g. 00, 01, 02 . . .)
numTrans	Number of Transactions
schemeLabel	Scheme Labe; 36 bytes maximum
RID	Registered Application Provider ID; 11 bytes maximum
csn	CSN; 32 bytes maximum
rec	EMV Table Record Number
csnFile	CSN File; 31 bytes maximum

Return values

Result	Code: • 00: Successful • 02: Invalid Data Format. One or more packets are invalid. • 04: Error Opening Table • 05: Error Reading Table • 06: Error Writing to a Table • 07: Error Closing Table • 08: Error in table record. The file containing the table data is corrupt or has invalid data. • 09: Table filename duplication. The table filename is already registered and has an associated table ID.

10.36.2.214 - (int) validateMacResponse: (NSData ∗) *macBlock* language:(int) *lcode* pinKey:(NSData ∗) *tpkKey* macKey:(NSData ∗) *takKey* msgLine1:(NSString ∗) *msg1* msgLine2:(NSString ∗) *msg2* macData:(NSData ∗) *data*

Perform Interac Response Analysis.

Validates the response MAC and displays any authorization messages returned by the host.

Upon successful execution, response can be obtained from property vfiEncryptionData.

XPI reference: S71

Parameters

macBlock	8-character MAC Block to verify
lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
tpkKey	PIN (TPK) Key
takKey	MAC (TAK) Key
msg1	Message Line 1 (0-16 bytes)
msg2	Message Line 2 (0-16 bytes)

data	Buffer to hold all data to be MAC'ed

Return values

Result	Code: • 00: Successful • 01: Unsuccessful • 05: Wrong Prefix (if balance is used, and failed) • 07: MAC did not verify • 09: No MAC block • 11: Bad Key • 14: Master key load failed. • 16: New Working Key load failed. Revert successfully to the previous Working Keys set. • 21: Cannot calculate MAC • 23: Invalid field length • 98: New Working Keys load failed. Cannot revert to the previous Working Keys

10.36.2.215 - (int) validateMacResponseFields: (NSData ∗) *macBlock* language:(int) *lcode* msgLine1:(NSString ∗) *msg1* msgLine2:(NSString ∗) *msg2* macData:(NSData ∗) *macdata* identity1:(NSData ∗) *identity1* data1:(NSData ∗) *data1* identityK:(NSData ∗) *identityK* dataK:(NSData ∗) *dataK*

Perform Interac Response Analysis.

Validates the response MAC and displays any authorization messages returned by the host.

Upon successful execution, response can be obtained from property vfiEncryptionData.

XPI reference: S71

Parameters

macBlock	8-character MAC Block to verify
lcode	Numeric value denoting the language code for determining the displayed language of the predefined prompt; NOTE: Any non-zero value will default to French.
msg1	Message Line 1 (0-16 bytes)
msg2	Message Line 2 (0-16 bytes)
macdata	Buffer to hold all data to be MAC'ed
identity1	Contains Base 24 field IDs (F, I, H, M) used to identify data in data1
data1	Based upon identity1, I - Working MSG Key, H - Working MAC Key, M - Working PIN Key. Value of this string is concatenated, in the sequence these fields are received, and used for MAC calculation. If working keys are received, they are stored accordingly.

identityK	Contains Base 24 field IDs (F, I, H, M) used to identify data in dataK
dataK	Based upon identity1, I - Working MSG Key, H - Working MAC Key, M - Working PIN Key. Value of this string is concatenated, in the sequence these fields are received, and used for MAC calculation. If working keys are received, they are stored accordingly.

Return values

Result	Code: • 00: Successful • 01: Unsuccessful • 05: Wrong Prefix (if balance is used, and failed) • 07: MAC did not verify • 09: No MAC block • 11: Bad Key • 14: Master key load failed. • 16: New Working Key load failed. Revert successfully to the previous Working Keys set. • 21: Cannot calculate MAC • 23: Invalid field length • 98: New Working Keys load failed. Cannot revert to the previous Working Keys

10.36.2.216 - (void) waitForResponse: (BOOL) *wait*

Toggles API method calls blocking for next command.

By default, calls are blocked while they wait for e210/e255/e315 to provide a response. Calling this method will turn off blocking on the next API call. The current value for this flag can be polled by evaluating read-only property willWaitFor-Response.

Parameters

wait	Amount of time to wait for account entry entry in response to manual PAN entry request before timeout • `TRUE` Wait for response (default behavior) • `FALSE` Do NOT wait for response on text command.

10.36.2.217 - (int) writeToScriptFile: (int) *scriptId* clear:(BOOL) *flag* numScripts:(int) *num* length:(int) *scriptLen* data:(NSData ∗) *scriptData*

Write to Script File.

This command allows the creation of EMVS71.DAT and EMVS72.DAT files adding the scripts received from the host to the inserted card. Currently, only one script can be sent at a time.

XPI reference: C25

Parameters

scriptId	Numeric '71' or '72' value to be set based on the script sent from the host.
flag	Indicator to note whether or not to clear the script file before writing; a value of YES will clear before writing
num	The number of scripts included in the request
scriptLen	Length of the script data
scriptData	Hexadecimal script data of length scriptLen sent from the host for writing to the inserted card.

Return values

Result	Code: • 00: Successful • 01: Unsuccessful

10.36.2.218 - (void) writeToSock: (NSData ∗) *data*

INTERNAL USE ONLY: Allows access for VFIBTBridge calls.

Do not call this method directly. This is exposed for VFIBTBridge to pass data to when communicating with e255

10.36.2.219 - (void) Z2: (NSString ∗) *string* clearScreen:(BOOL) *clear*

Display a String.

Deprecated Use displayMessages:message1:message2:clearScreen:

10.36.2.220 - (void) Z3: (int) *numMessages* message1:(NSString ∗) *msg1* message2:(NSString ∗) *msg2* clearScreen:(BOOL) *clear*

Display Messages.

See Also

- displayMessages:message1:message2:clearScreen:

10.36.2.221 - (int) Z50: (int) *echoFlag* timeout:(int) *timeout* maxLen:(int) *length*

Request String Input.

See Also

- requestStringInput:timeout:maxLen:

10.36.2.222 - (int) Z50: (int) *echoFlag* timeout:(int) *timeout* maxLen:(int) *length* entryType:(int) *type*

Request String Input.

See Also

- requestStringInput:timeout:maxLen:entryType:

10.36.2.223 - (int) Z52: (NSString ∗) *message* echo:(int) *echoFlag* timeout:(int) *timeout* maxLen:(int) *length* entryType:(int) *type*

Display Message and Prompt for Input.

See Also

- displayMessagePromptInput:echo:timeout:maxLen:entryType:

10.36.2.224 - (int) Z56: (int) *amount* message1:(NSString ∗) *msg1* message2:(NSString ∗) *msg2*

Amount Entry.

See Also

- amountEntry:message1:message2:

10.36.2.225 - (int) Z58: (int) *defaultValue* amountFormat:(int) *format* message1:(NSString ∗) *msg1* message2:(NSString ∗) *msg2*

Amount Introduction.

See Also

- amountIntroduction:amountFormat:message1:message2:

10.36.2.226 - (int) Z60: (NSString ∗) *accountNumber*

Accept and Encrypt PIN.

See Also

- acceptEncryptPIN:

10.36.2.227 - (int) Z62: (NSString ∗) *accountNumber* minPIN:(int) *min* maxPIN:(int) *max* requirePIN:(BOOL) *req* firstMessage:(NSString ∗) *msg1* secondMessage:(NSString ∗) *msg2* processingMessage:(NSString ∗) *procMsg*

Accept and Encrypt PIN, Display Custom Messages.

See Also

- acceptEncryptPINMessage:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processingMessage:

10.36.3 Property Documentation

10.36.3.1 - (int) accountEntryTimeout `[read],[atomic],[assign]`

Read Only Timeout Value waiting for account entry during EMV transaction. Set with accountEntryTimeout:()

10.36.3.2 - (int) ackTimeout [read], [atomic], [assign]

Read Only Timeout Value waiting for pinPad to return an <ACK> from an executed command. Set with setResponseTimeout:()

10.36.3.3 - (int) amountResponse [read], [atomic], [assign]

Read Only value returned with successful response code to amountIntroduction:format:message1:message2:timeout:()

10.36.3.4 - (BOOL) BTconnected [read], [atomic], [assign]

Read Only Boolean pinpad e210/e255/e315 Bluetooth connection Status.

10.36.3.5 - (BOOL) connected [read], [atomic], [assign]

Read Only Boolean pinpad connection status.

10.36.3.6 - (NSMutableArray*) connectedSockets [read], [write], [nonatomic], [retain]

PRIVATE: Used by VMF to manage TCP Server Mode.

10.36.3.7 - (int) continueTransactionTimeout [read], [atomic], [assign]

Read Only Timeout Value waiting to complete online EMV transaction. Set with continueTransactionTimeout:()

10.36.3.8 - (AsyncSocket*) currentSocket [read], [write], [nonatomic], [retain]

PRIVATE: Used by VMF to manage TCP Server Mode.

10.36.3.9 - (id< VFIPinpadDelegate >) delegate [read], [write], [atomic], [retain]

Gets or Sets delegate for protocols.

10.36.3.10 - (EAAccessory*) eaACC [read], [nonatomic], [assign]

Read Only Connected Accessory.

10.36.3.11 - (NSArray*) emaFiles [read], [nonatomic], [assign]

Array of VFIEMAFiles containing the response from executing requestEMAFiles

10.36.3.12 - (NSData*) emvTagData [read], [nonatomic], [assign]

Read Only value returned with successful response code to amountIntroduction:format:message1:message2:timeout:()

10.36.3.13 - (int) frameworkTimeout [read], [atomic], [assign]

Read Only Timeout Value waiting for pinPad to return a reply from an executed command. Set with setFramework-Timeout:

10.36.3.14 - (BOOL) initialized [read], [atomic], [assign]

Read Only Boolean pinpad initialized.

10.36.3.15 - (int) ioTimer [read], [atomic], [assign]

Read Only Timeout Value XPI awaiting input/output. Set with setIOTimer:

10.36.3.16 - (BOOL) isGen3 [read], [atomic], [assign]

Read Only Boolean Gen3 Connected.

10.36.3.17 - (BOOL) isRunning [read], [write], [atomic]

PRIVATE: Used by VMF to manage TCP Server Mode.

10.36.3.18 - (BOOL) isVMFRemote [read], [write], [atomic]

PRIVATE: Used by VMF to manage VMF Remote Mode.

10.36.3.19 - (NSString*) keyLoadErrorCode [read], [nonatomic], [assign]

Error code returned from unsuccessful Key Loading.

Return Values:

- 00000000: Successful PKI update.
- 00001000: Parameter Error. NULL or out of Range.
- 00001001: Invalid Public Key File.
- 00001002: Encryption Failed.
- 00001003: Too small output buffer.
- 00001004: Invalid Key ID File.
- 00001005: Error in opening DataFile.
- 00001006: Error in Hash Key computation.
- 00001100: Unknown or unexpected error.

10.36.3.20 - (AsyncSocket*) listenSocket [read], [write], [nonatomic], [retain]

PRIVATE: Used by VMF to manage TCP Server Mode.

10.36.3.21 - (BOOL) pinpadConnected [read], [atomic], [assign]

Read Only Boolean pinpad connection status.

10.36.3.22 - (NSString*) pinpadFirmwareRevision [read], [nonatomic], [assign]

Read Only Firmware Version reported to External Accessory.

10.36.3.23 - (NSString*) pinpadHardwareRevision [read], [nonatomic], [assign]

Read Only Hardware Version reported to External Accessory.

10.36.3.24 - (NSString*) pinpadManufacturer [read], [nonatomic], [assign]

Read Only name reported to External Accessory.

10.36.3.25 - (NSString*) pinpadModelNumber [read], [nonatomic], [assign]

Read Only Manufacturer reported to External Accessory.

10.36.3.26 - (NSString*) pinpadName [read], [nonatomic], [assign]

Gets or Sets delegate for protocols.

10.36.3.27 - (NSString*) pinpadSerialNumber [read], [nonatomic], [assign]

Read Only Serial Number reported to External Accessory.

10.36.3.28 - (int) PINTimeout [read], [atomic], [assign]

Read Only Timeout Value waiting for PIN entry during EMV transaction. Set with setPINTimeout:

10.36.3.29 - (BOOL) remoteMode [read], [write], [atomic]

PRIVATE: Used by VMF to manage VMF Remote Mode.

10.36.3.30 - (BOOL) serverRunning [read], [atomic], [assign]

Read Only Boolean TCP server status.

10.36.3.31 - (VFICardData*) vfiCardData [read], [write], [nonatomic], [retain]

Holds card data results from executing obtainCardData:, obtainTrack2Data:optionalAmount:, or getCardData:language-:amount:otherAmount:

10.36.3.32 - (VFICipheredData*) vfiCipheredData [read], [write], [nonatomic], [retain]

Holds cyphered results from executing retrieveEParmsData, performTGKUpdateRequest, getPKICipheredData, getCipheredData

10.36.3.33 - (VFIDataEntry*) vfiDataEntry [read], [write], [nonatomic], [retain]

Holds prompt results from executing S21prompt:, dataEntryRequest:language:tip:cashback:surcharge:, combinedPromptCommand:language:transID:menuArray:surcharge:accountNum:tipOptions:cashBackOptions:surchargeOptions:

10.36.3.34 - (VFIDiagnostics*) vfiDiagnostics [read], [write], [nonatomic], [retain]

Holds result from executing diagnosticInfo

10.36.3.35 - (VFIEMVAuthorization*) vfiEMVAuthorization [read], [write], [nonatomic], [retain]

Holds result from executing cardAuthorization:otherAmount:merchantDecision:minRequestObjects:

10.36.3.36 - (VFIEMVCompletionData*) vfiEMVCompletionData [read], [write], [nonatomic], [retain]

Holds result from executing completeOnlineEMV:displayResult:hostTags:

10.36.3.37 - (VFIEMVTags*) vfiEMVTags [read], [write], [nonatomic], [retain]

Holds prompt results from executing getEMVTags:

10.36.3.38 - (VFIEncryptionData*) vfiEncryptionData [read], [write], [nonatomic], [retain]

Holds result from executing testPINPad, createMac:, doInteracRequest:language:amount:pan:mac:, validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData:, acceptEncryptPIN:, acceptEncryptPINMessage:minPIN:maxPIN:requirePIN:firstMessage:secondMessage:processingMessage:

10.36.3.39 - (int) vsdKeyIndex [read], [atomic], [assign]

Read Only Index of DUKPT key when VSD encryption is enabled. Set after calling setPINTimeout:

The documentation for this class was generated from the following file:

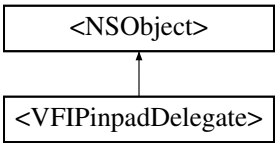
- VFIPinpad.h

10.37 <VFIPinpadDelegate> Protocol Reference

Protocol methods established for VFIPinpad class NEW.

```
#import <VFIPinpad.h>
```

Inheritance diagram for <VFIPinpadDelegate>:



Instance Methods

- (void) - pinpadResultCode:result:

  Result code received.
- (void) - pinpadLogEntry:withSeverity:

  When logEnabled: (VFIPinpad) is passed `TRUE`*, this delegate will receive log entries.*
- (void) - pinpadSerialData:incoming:

  All incoming/outgoing data going to the XPI application can be monitored through this delegate.
- (void) - btSerialData:incoming:

  All incoming/outgoing data going to the bluetooth interface can be monitored through this delegate.
- (void) - pinpadServerData:incoming:

  All incoming/outgoing data going to the TCP server from a remote socket can be monitored through this delegate.
- (void) - pinpadMSRData:expMonth:expYear:trackData:

  Receives card data from MSR swipe activated by enableMSR (VFIPinpad) .
- (void) - pinpadMSRData:expMonth:expYear:track1Data:track2Data:

  Receives card data from MSR swipe activated by enableMSRDualTrack (VFIPinpad) .
- (void) - pinpadDownloadInfo:

  Receives informational messages during process of downloading data to e210/e255/e315 .
- (void) - pinpadDownloadBlocks:sent:

  Receives transferred data statistics while downloading data to e210/e255/e315.
- (void) - pinPadInitialized:

  Notifies of initialization state changes in e210/e255/e315 XPI Application.
- (void) - pinpadReconnectStarted

  When the External Accessory reports the e210/e255/e315 is detected (but not initialized by framework),this delegate is called. This signifies the beginning of the framework initialization process of the XPI application.
- (void) - pinpadReconnectFinished

  This signifies the end of the framework initialization process for XPI application.
- (void) - pinpadConnected:

  Notified of XPI connection/disconnection events.
- (void) - serverConnected:

  Notified of TCP Server connection/disconnection events.
- (void) - pinpadDataReceived:

  This delegate monitors all data received from XPI Application.
- (void) - pinpadDataSent:

  This delegate monitors all data sent from XPI Application.
- (void) - pinpadMessage:

10.37.1 Detailed Description

Protocol methods established for VFIPinpad class NEW.

10.37.2 Method Documentation

10.37.2.1 - (void) btSerialData: (NSData *) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the bluetooth interface can be monitored through this delegate.

This data will be process and further reported to individual serialData delegates

Parameters

data	The serial data represented as a NSData object
isIncoming	The direction of the data • `TRUE` specifies data being received from Bluetooth interface, • `FALSE` indicates data being sent to bluetooth interface.

10.37.2.2 - (void) pinpadConnected: (BOOL) *isConnected* [optional]

Notified of XPI connection/disconnection events.

A connect/disconnect can either be from a physical disconnection/connection with the External Accessory API, or from an application going to backround or returning to foreground.

Parameters

isConnected	A new connection or disconnection was detected • `TRUE` The XPI application has connected. • `FALSE` The XPI application has disconnected.

10.37.2.3 - (void) pinpadDataReceived: (NSData *) *data* [optional]

This delegate monitors all data received from XPI Application.

Parameters

data	A NSData binary object representing the incoming data from XPI.

10.37.2.4 - (void) pinpadDataSent: (NSData *) *data* [optional]

This delegate monitors all data sent from XPI Application.

Parameters

data	A NSData binary object representing the outgoing data sent to XPI.

10.37.2.5 - (void) pinpadDownloadBlocks: (int) *TotalBlocks* sent:(int) *BlocksSent* [optional]

Receives transferred data statistics while downloading data to e210/e255/e315.

Parameters

TotalBlocks	Total file size being transferred
BlocksSent	Amount of data successfully sent

10.37.2.6 - (void) pinpadDownloadInfo: (NSString ∗) *log* [optional]

Receives informational messages during process of downloading data to e210/e255/e315 .

Parameters

log	Message generated by framework during the download process to either signify status or error at various stages

10.37.2.7 - (void) pinPadInitialized: (BOOL) *isInitialized* [optional]

Notifies of initialization state changes in e210/e255/e315 XPI Application.

Parameters

isInitialized	Device initialization state change • `TRUE` successfully initialized the XPI application, • `FALSE` XPI application went offline.

10.37.2.8 - (void) pinpadLogEntry: (NSString ∗) *logEntry* withSeverity:(int) *severity* [optional]

When logEnabled: (VFIPinpad) is passed `TRUE`, this delegate will receive log entries.

Parameters

logEntry	The log entry
severity	The severitry of the log entry, with 0 indicating highest priority. Uses enumeration SEVERITY_LOG_LEVELS

10.37.2.9 - (void) pinpadMessage: (NSString ∗) *message* [optional]

10.37.2.10 - (void) pinpadMSRData: (NSString ∗) *pan* expMonth:(NSString ∗) *month* expYear:(NSString ∗) *year* track1Data:(NSString ∗) *track1* track2Data:(NSString ∗) *track2* [optional]

Receives card data from MSR swipe activated by enableMSRDualTrack (VFIPinpad) .

Parameters

pan	Account Number (masked or encoded for security)
month	Card Expiration Month

year	Card Expiration Year
track1	Complete track 1 data (encoded for security)
track2	Complete track 2 data (encoded for security)

10.37.2.11 - (void) pinpadMSRData: (NSString ∗) *pan* expMonth:(NSString ∗) *month* expYear:(NSString ∗) *year* trackData:(NSString ∗) *track2* [optional]

Receives card data from MSR swipe activated by enableMSR (VFIPinpad) .

Parameters

pan	Account Number (masked or encoded for security)
month	Card Expiration Month
year	Card Expiration Year
track2	Complete track 2 data (encoded for security)

10.37.2.12 - (void) pinpadReconnectFinished [optional]

This signifies the end of the framework initialization process for XPI application.

10.37.2.13 - (void) pinpadReconnectStarted [optional]

When the External Accessory reports the e210/e255/e315 is detected (but not initialized by framework),this delegate is called. This signifies the beginning of the framework initialization process of the XPI application.

10.37.2.14 - (void) pinpadResultCode: (NSString ∗) *command* result:(int) *code* [optional]

Result code received.

Receives int value representing the returned result code after executing a API call represented by `command`.

Parameters

code	Result code
command	XPI command executed

10.37.2.15 - (void) pinpadSerialData: (NSData ∗) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the XPI application can be monitored through this delegate.

Parameters

data	The serial data represented as a NSData object
isIncoming	The direction of the data • `TRUE` specifies data being received from XPI application, • `FALSE` indicates data being sent to XPI application.

10.37.2.16 - (void) pinpadServerData: (NSData *) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the TCP server from a remote socket can be monitored through this delegate.

Parameters

data	The serial data represented as a NSData object
isIncoming	The direction of the data • `TRUE` specifies data being received by the TCP server for sending to XPI, • `FALSE` indicates data being sent to remote socket.

10.37.2.17 - (void) serverConnected: (BOOL) *isConnected* `[optional]`

Notified of TCP Server connection/disconnection events.

A connect/disconnect can either be from a physical disconnection/connection with the External Accessory API, or from an application going to backround or returning to foreground.

Parameters

isConnected	A new connection or disconnection was detected • `TRUE` The TCP Server has connected. • `FALSE` The TCP Server has disconnected.

The documentation for this protocol was generated from the following file:

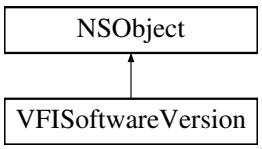
- VFIPinpad.h

10.38 VFISoftwareVersion Class Reference

Encapsulating data class utilized by VFIControl.

`#import <VFISoftwareVersion.h>`

Inheritance diagram for VFISoftwareVersion:



Instance Methods

- (void) - clear

  clears all VFISoftwareVersion properties

Properties

- NSString * AppMajor

  Software Version - Major.

- NSString * AppMinor

  Software Version - Minor.
- NSString * AppBuild

  Software Version - Build.
- NSString * OSPlatform

  Software OS Platform.
- NSString * OSID

  Software Identification.
- NSString * OSVersion

  Software Version - Main.
- NSString * OSSubVersion

  Software Version - Sub.
- NSString * OSBuildYear

  Software Version - Build Year.
- NSString * OSBuildMonth

  Software Version - Build Month.
- NSString * OSBuildDay

  Software Version - Build Day.

10.38.1 Detailed Description

Encapsulating data class utilized by VFIControl.

10.38.2 Method Documentation

10.38.2.1 - (void) clear

clears all VFISoftwareVersion properties

10.38.3 Property Documentation

10.38.3.1 - (NSString *) AppBuild [read], [write], [nonatomic], [retain]

Software Version - Build.

10.38.3.2 - (NSString *) AppMajor [read], [write], [nonatomic], [retain]

Software Version - Major.

10.38.3.3 - (NSString *) AppMinor [read], [write], [nonatomic], [retain]

Software Version - Minor.

10.38.3.4 - (NSString *) OSBuildDay [read], [write], [nonatomic], [retain]

Software Version - Build Day.

10.38.3.5 - (NSString *) OSBuildMonth [read], [write], [nonatomic], [retain]

Software Version - Build Month.

10.38.3.6 - (NSString *) OSBuildYear [read], [write], [nonatomic], [retain]

Software Version - Build Year.

10.38.3.7 - (NSString *) OSID [read], [write], [nonatomic], [retain]

Software Identification.

10.38.3.8 - (NSString *) OSPlatform [read], [write], [nonatomic], [retain]

Software OS Platform.

10.38.3.9 - (NSString *) OSSubVersion [read], [write], [nonatomic], [retain]

Software Version - Sub.

10.38.3.10 - (NSString *) OSVersion [read], [write], [nonatomic], [retain]

Software Version - Main.

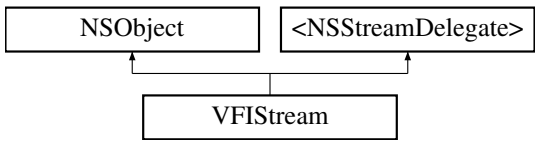
The documentation for this class was generated from the following file:

- VFISoftwareVersion.h

10.39 VFIStream Class Reference

`#import <VFIStream.h>`

Inheritance diagram for VFIStream:



Instance Methods

- (void) - setupControllerForAccessory:withProtocolString:
- (BOOL) - openSession
- (void) - closeSession
- (void) - writeData:channel:isOpen:
- (void) - setBTSessionOpen:

- (BOOL) - isGen3
- (BOOL) - isBT
- (NSUInteger) - readBytesAvailable
- (NSData ∗) - readData:

Protected Attributes

- EAAccessory ∗ _accessory
- EASession ∗ _session
- NSString ∗ _protocolString
- NSMutableData ∗ _writeData
- NSMutableData ∗ _readData
- bool _sessionOpen
- bool _sessionOpen_XPI
- bool _sessionOpen_Barcode
- bool _sessionOpen_Control
- bool _sessionOpen_Zontalk

Properties

- bool sessionOpen
- bool sessionOpen_XPI
- bool sessionOpen_Barcode
- bool sessionOpen_Control
- bool sessionOpen_Zontalk
- EAAccessory ∗ accessory
- NSString ∗ protocolString
- id< VFIStreamDelegate > delegate

10.39.1 Method Documentation

10.39.1.1 - (void) closeSession

10.39.1.2 - (BOOL) isBT

10.39.1.3 - (BOOL) isGen3

10.39.1.4 - (BOOL) openSession

10.39.1.5 - (NSUInteger) readBytesAvailable

10.39.1.6 - (NSData ∗) readData: (NSUInteger) *bytesToRead*

10.39.1.7 - (void) setBTSessionOpen: (int) *BTChannel*

10.39.1.8 - (void) setupControllerForAccessory: (EAAccessory ∗) *accessory* withProtocolString:(NSString ∗) *protocolString*

10.39.1.9 - (void) writeData: (NSData ∗) *data* channel:(int) *channel* isOpen:(BOOL) *isOpen*

10.39.2 Member Data Documentation

10.39.2.1 - (EAAccessory*) _accessory [protected]

10.39.2.2 - (NSString*) _protocolString [protected]

10.39.2.3 - (NSMutableData*) _readData [protected]

10.39.2.4 - (EASession*) _session [protected]

10.39.2.5 - (bool) _sessionOpen [protected]

10.39.2.6 - (bool) _sessionOpen_Barcode [protected]

10.39.2.7 - (bool) _sessionOpen_Control [protected]

10.39.2.8 - (bool) _sessionOpen_XPI [protected]

10.39.2.9 - (bool) _sessionOpen_Zontalk [protected]

10.39.2.10 - (NSMutableData*) _writeData [protected]

10.39.3 Property Documentation

10.39.3.1 - (EAAccessory*) accessory [read], [nonatomic], [assign]

10.39.3.2 - (id< VFIStreamDelegate >) delegate [read], [write], [atomic], [retain]

10.39.3.3 - (NSString*) protocolString [read], [nonatomic], [assign]

10.39.3.4 - (bool) sessionOpen [read], [atomic], [assign]

10.39.3.5 - (bool) sessionOpen_Barcode [read], [atomic], [assign]

10.39.3.6 - (bool) sessionOpen_Control [read], [atomic], [assign]

10.39.3.7 - (bool) sessionOpen_XPI [read], [atomic], [assign]

10.39.3.8 - (bool) sessionOpen_Zontalk [read], [atomic], [assign]

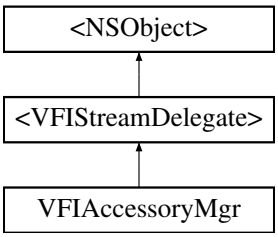
The documentation for this class was generated from the following file:

- VFIStream.h

10.40 <VFIStreamDelegate> Protocol Reference

```
#import <VFIStream.h>
```

Inheritance diagram for <VFIStreamDelegate>:



Instance Methods

- (void) - receivedData:protocol:

10.40.1 Method Documentation

10.40.1.1 - (void) receivedData: (NSObject *) *data* protocol:(NSString *) *protocol* [optional]

The documentation for this protocol was generated from the following file:

- VFIStream.h

Chapter 11

File Documentation

11.1 AsyncSocket.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- protocol <AsyncSocketDelegate>
- class AsyncSocket

Typedefs

- typedef enum AsyncSocketError AsyncSocketError

Enumerations

- enum AsyncSocketError {
  AsyncSocketCFSocketError = kCFSocketError, AsyncSocketNoError = 0, AsyncSocketCanceledError, AsyncSocketConnectTimeoutError,
  AsyncSocketReadMaxedOutError, AsyncSocketReadTimeoutError, AsyncSocketWriteTimeoutError }

Variables

- NSString *const AsyncSocketException
- NSString *const AsyncSocketErrorDomain

11.1.1 Typedef Documentation

11.1.1.1 typedef enum AsyncSocketError AsyncSocketError

11.1.2 Enumeration Type Documentation

11.1.2.1 enum AsyncSocketError

Enumerator

AsyncSocketCFSocketError

AsyncSocketNoError

AsyncSocketCanceledError

AsyncSocketConnectTimeoutError

AsyncSocketReadMaxedOutError

AsyncSocketReadTimeoutError

AsyncSocketWriteTimeoutError

11.1.3 Variable Documentation

11.1.3.1 NSString* const AsyncSocketErrorDomain

11.1.3.2 NSString* const AsyncSocketException

11.2 Barcode_host.h File Reference

Classes

- struct T_BEEP
- struct T_BEEP_DEF
- struct M_BEEP_DEF
- struct PARM_DEF
- struct CMD_DEF
- struct HOST_TX_BUF
- struct HOST_RX_BUF
- struct symb_INFO

Macros

- #define HOST_BUF_SIZE 2048
- #define CMD_BAR_DEV_OPEN 0x1A
- #define CMD_BAR_DEV_CLOSE 0x1B
- #define CMD_START_SCAN 0x01
- #define CMD_STOP_SCAN 0x02
- #define CMD_PASS_THRU 0x03
- #define CMD_SINGLE_GET_APP_VER 0x04
- #define CMD_SET_TRIG_MODE 0x05
- #define CMD_BEEP_IMMEDIATE 0x06
- #define CMD_SINGLE_BUTTON_STATUS 0x07
- #define CMD_SINGLE_GET_FIRM_VER 0x08
- #define CMD_SINGLE_GET_DEVICE_ID 0x09
- #define CMD_MULTI_SCAN 0x0A
- #define CMD_RESTORE_DEFAULTS 0x0B
- #define CMD_EN_PICKLIST_MODE 0x0C
- #define CMD_SCAN_TIMEOUT 0x0D

- #define CMD_TIMEOUT_BW_SAME_SYM 0x0E
- #define CMD_TIMEOUT_BW_DIFF_SYM 0x0F
- #define CMD_EN_CONTINUOUS_RD 0x10
- #define CMD_EN_UNIQUE_CODE_REPO 0x11
- #define CMD_EN_MOBILE_PH_MODE 0x12
- #define CMD_PREFIX_VAL 0x13
- #define CMD_SUFFIX1_VAL 0x14
- #define CMD_SUFFIX2_VAL 0x15
- #define CMD_EN_XMIT_FMT 0x16
- #define CMD_EN_AIM_PATTERN 0x17
- #define CMD_SYMBOLOGY 0x18
- #define CMD_DISABLE_ALL_SYMB 0x19
- #define CMD_AUTO_BEEP_CONFIG 0x1C
- #define CMD_GET_AUTO_BEEP_CONFIG 0x1D
- #define SYM_PID_EN_CODE128 0x00
- #define SYM_PID_SETLEN_1DISCRETE_C128 0x01
- #define SYM_PID_SETLEN_2DISCRETE_C128 0x02
- #define SYM_PID_SETLEN_RANGE_C128 0x03
- #define SYM_PID_SETLEN_ANY_C128 0x04
- #define SYM_PID_EN_GS1128 0x05
- #define SYM_PID_EN_ISBT 0x06
- #define SYM_PID_ISBT_CONCATE 0x07
- #define SYM_PID_CHECK_ISBT_TABLE 0x08
- #define SYM_PID_ISBT_CONCATE_REDUN 0x09
- #define SYM_PID_EN_UPCA 0x0A
- #define SYM_PID_EN_UPCE 0x0B
- #define SYM_PID_EN_UPCE1 0x0C
- #define SYM_PID_EN_EAN8_JAN8 0x0D
- #define SYM_PID_EN_EAN13_JAN13 0x0E
- #define SYM_PID_EN_BOOKLAND_EAN 0x0F
- #define SYM_PID_EN_ISBN_FORMAT 0x10
- #define SYM_PID_SUPPLIMENTS_UPC_EAN 0x11
- #define IGNORE_UPC_EAN_W_SUPPLIMENTS ∗0x00
- #define DECODE_UPC_EAN_W_SUPPLIMENTS 0x01
- #define AUTODESCRIMINATE_SUPPLIMENTS 0x02
- #define ENABLE_378_379_SUPPLIMENTS 0x04
- #define ENABLE_978_979_SUPPLIMENTS 0x05
- #define ENABLE_977_SUPPLIMENT 0x07
- #define ENABLE_414_419_434_439 0x06
- #define ENABLE_491_SUPPLIMENTS 0x08
- #define ENABLE_SMART_SUPPLIMENT 0x03
- #define SUPPLIMENT_USER_PROG1 0x09
- #define SUPPLIMENT_USER_PROG1_2 0x0A
- #define SMART_SUPPLIMENT_USER_PROG1 0x0B
- #define SMART_SUPPLIMENT_USER_PROG1_2 0x0C
- #define SYM_PID_USER_PROG_SUPP1 0x12
- #define SYM_PID_USER_PROG_SUPP2 0x13
- #define SYM_PID_UPC_EAN_JAN_SUPP_REDUN 0x14
- #define SYM_PID_XMIT_UPCA_CHECK_DIGIT 0x15
- #define SYM_PID_XMIT_UPCE_CHECK_DIGIT 0x16
- #define SYM_PID_XMIT_UPCE1_CHECK_DIGIT 0x17

- #define SYM_PID_XMIT_UPCA_PREAMBLE 0x18
- #define SYM_PID_XMIT_UPCE_PREAMBLE 0x19
- #define SYM_PID_XMIT_UPCE1_PREAMBLE 0x1A
- #define SYM_PID_CONVERT_UPCE_2_UPCA 0x1B
- #define SYM_PID_CONVERT_UPCE1_2_UPCA 0x1C
- #define SYM_PID_EAN8_JAN8_EXTEND 0x1D
- #define SYM_PID_UCC_COUPON_EXTEND 0x1E
- #define SYM_PID_COUPON_REPORT 0x1F
- #define SYM_PID_ISSN_EAN 0x20
- #define SYM_PID_EN_CODE39 0x21
- #define SYM_PID_EN_TRIOPTIC_CODE39 0x22
- #define SYM_PID_CONV_CODE39_2_CODE32 0x23
- #define SYM_PID_CODE32_PREFIX 0x24
- #define SYM_PID_SETLEN_1DISCRETE_C39 0x25
- #define SYM_PID_SETLEN_2DISCRETE_C39 0x26
- #define SYM_PID_SETLEN_RANGE_C39 0x27
- #define SYM_PID_SETLEN_ANY_C39 0x28
- #define SYM_PID_CODE39_CHK_DIGIT 0x29
- #define SYM_PID_XMIT_CODE39_CHK_DIGIT 0x2A
- #define SYM_PID_CODE39_FULL_ASCII 0x2B
- #define SYM_PID_CODE39_BUFFERING 0x2C
- #define SYM_PID_EN_INTER2OF5 0x2D
- #define SYM_PID_SETLEN_1DISCRETE_I2OF5 0x2E
- #define SYM_PID_SETLEN_2DISCRETE_I2OF5 0x2F
- #define SYM_PID_SETLEN_RANGE_I2OF5 0x30
- #define SYM_PID_SETLEN_ANY_I2OF5 0x31
- #define SYM_PID_I2OF5_CHECK_DIGIT 0x32
- #define SYM_PID_XMIT_I2OF5_CHECK_DIGIT 0x33
- #define SYM_PID_CONV_I2OF5_EAN13 0x34
- #define SYM_PID_EN_QR_CODE 0x35
- #define SYM_PID_EN_QR_INVERSE 0x36
- #define SYM_PID_EN_MICRO_QR 0x37
- #define SYM_PID_EN_DATA_MATRIX 0x38
- #define SYM_PID_EN_DATA_MATRIX_INVERSE 0x39
- #define SYM_PID_EN_MIRROR_IMAGES 0x3A
- #define SYM_PID_EN_CODE93 0x3B
- #define SYM_PID_SETLEN_1DISCRETE_C93 0x3C
- #define SYM_PID_SETLEN_2DISCRETE_C93 0x3D
- #define SYM_PID_SETLEN_RANGE_C93 0x3E
- #define SYM_PID_SETLEN_ANY_C93 0x3F
- #define SYM_PID_EN_CODE11 0x40
- #define SYM_PID_SETLEN_1DISCRETE_C11 0x41
- #define SYM_PID_SETLEN_2DISCRETE_C11 0x42
- #define SYM_PID_SETLEN_RANGE_C11 0x43
- #define SYM_PID_SETLEN_ANY_C11 0x44
- #define SYM_PID_CODE11_CHK_DIGIT 0x45
- #define SYM_PID_XMIT_CODE11_CHK_DIGIT 0x46
- #define SYM_PID_EN_DISC2OF5 0x47
- #define SYM_PID_SETLEN_1DISCRETE_D2OF5 0x48
- #define SYM_PID_SETLEN_2DISCRETE_D2OF5 0x49
- #define SYM_PID_SETLEN_RANGE_D2OF5 0x4A

- #define SYM_PID_SETLEN_ANY_D2OF5 0x4B
- #define SYM_PID_EN_CODABAR 0x4C
- #define SYM_PID_SETLEN_1DISCRETE_CBAR 0x4D
- #define SYM_PID_SETLEN_2DISCRETE_CBAR 0x4E
- #define SYM_PID_SETLEN_RANGE_CBAR 0x4F
- #define SYM_PID_SETLEN_ANY_CBAR 0x50
- #define SYM_PID_EN_CLSI 0x51
- #define SYM_PID_EN_NOTIS 0x52
- #define SYM_PID_START_STOP_CASE 0x53
- #define SYM_PID_EN_MSI 0x54
- #define SYM_PID_SETLEN_1DISCRETE_MSI 0x55
- #define SYM_PID_SETLEN_2DISCRETE_MSI 0x56
- #define SYM_PID_SETLEN_RANGE_MSI 0x57
- #define SYM_PID_SETLEN_ANY_MSI 0x58
- #define SYM_PID_MSI_CHK_DIGIT 0x59
- #define SYM_PID_XMIT_MSI_CHK_DIGIT 0x5A
- #define SYM_PID_MSI_CHK_DIGIT_ALGOR 0x5B
- #define SYM_PID_EN_CHINESE2OF5 0x5C
- #define SYM_PID_EN_MATRIX2OF5 0x5D
- #define SYM_PID_SETLEN_1DISCRETE_M2OF5 0x5E
- #define SYM_PID_SETLEN_2DISCRETE_M2OF5 0x5F
- #define SYM_PID_SETLEN_RANGE_M2OF5 0x60
- #define SYM_PID_SETLEN_ANY_M2OF5 0x61
- #define SYM_PID_M2OF5_CHK_DIGIT 0x62
- #define SYM_PID_XMIT_M2OF5_CHK_DIGIT 0x63
- #define SYM_PID_EN_KOREAN3OF5 0x64
- #define SYM_PID_EN_INVERSE1D 0x65
- #define SYM_PID_EN_US_POSTNET 0x66
- #define SYM_PID_EN_US_PLANET 0x67
- #define SYM_PID_XMIT_US_POST_CHK_DIGIT 0x68
- #define SYM_PID_EN_UK_POSTAL 0x69
- #define SYM_PID_XMIT_UK_POST_CHK_DIGIT 0x6A
- #define SYM_PID_EN_JAPAN_POSTAL 0x6B
- #define SYM_PID_EN_AUSTRALIA_POST 0x6C
- #define SYM_PID_EN_AUST_POST_FMT 0x6D
- #define SYM_PID_EN_NETHERLANDS_KIX 0x6E
- #define SYM_PID_EN_USPS_4CB 0x6F
- #define SYM_PID_EN_UPU_FICS 0x70
- #define SYM_PID_EN_GS1_DATABAR 0x71
- #define SYM_PID_EN_GS1_LIMITED 0x72
- #define SYM_PID_LTD_SECURITY 0x73
- #define SYM_PID_EN_GS1_EXPANDED 0x74
- #define SYM_PID_EN_CONV_UPC_EAN 0x75
- #define SYM_PID_EN_COMP_CC_C 0x76
- #define SYM_PID_EN_COMP_CC_A_B 0x77
- #define SYM_PID_EN_COMP_TLC_39 0x78
- #define SYM_PID_UPC_COMP_MODE 0x79
- #define SYM_PID_EN_GS1_128_EMULATION 0x7A
- #define SYM_PID_EN_PDF417 0x7B
- #define SYM_PID_EN_MICRO_PDF417 0x7C
- #define SYM_PID_EN_MAXICODE 0x7D

- #define SYM_PID_EN_AZTEC 0x7E
- #define SYM_PID_EN_AZTEC_INVERSE 0x7F
- #define GEN_PID_PASS_THRU 0xFF
- #define GEN_PID_SET_TRIG_MODE 0xFE
- #define GEN_PID_BEEP_IMMEDIATE 0xFD
- #define GEN_PID_RESTORE_DEFAULTS 0xFC
- #define GEN_PID_PICKLIST_MODE 0xFB
- #define GEN_PID_SCAN_TIMEOUT 0xFA
- #define GEN_PID_TIMEOUT_BW_SAME_SYM 0xF9
- #define GEN_PID_TIMEOUT_BW_DIFF_SYM 0xF8
- #define GEN_PID_CONTINUOUS_READ 0xF7
- #define GEN_PID_UNIQUE_CODE_REPORT 0xF6
- #define GEN_PID_MOBILE_PHONE_MODE 0xF5
- #define GEN_PID_PREFIX_KEY 0xF4
- #define GEN_PID_PREFIX_VAL 0xF3
- #define GEN_PID_SUFFIX1_KEY 0xF2
- #define GEN_PID_SUFFIX1_VAL 0xF1
- #define GEN_PID_SUFFIX2_KEY 0xF0
- #define GEN_PID_SUFFIX2_VAL 0xEF
- #define GEN_PID_SCAN_DATA_XMIT_FMT 0xEE
- #define GEN_PID_AIM_PATTERN_EN 0xED
- #define GEN_PID_AUTO_BEEP_MODE 0xEC
- #define GEN_PID_AUTO_BEEP_SCAN 0xEB
- #define GEN_PID_AUTO_BEEP_ERROR 0xEA
- #define CMD_BARCODE_DATA 0x01
- #define RESP_ACK 0x00000000
- #define RESP_NAK 0x00000001
- #define RESP_BARCODE_DATA 0x80000000
- #define RESP_BUTTON_STATUS 0x80000001
- #define CODE_ID_UPC_EA 0x01
- #define CODE_ID_CODE39_32 0x02
- #define CODE_ID_CODABAR 0x03
- #define CODE_ID_CODE128_ISBT 0x04
- #define CODE_ID_CODE93 0x05
- #define CODE_ID_INTL2OF5 0x06
- #define CODE_ID_DISC2OF5 0x07
- #define CODE_ID_CODE11 0x08
- #define CODE_ID_MSI 0x09
- #define CODE_ID_GSI128 0x0A
- #define CODE_ID_BOOKLAND_EAN 0x0B
- #define CODE_ID_TRIOPTIC39 0x0C
- #define CODE_ID_COUPONCODE 0x0D
- #define CODE_ID_GS1DATABAR 0x0E
- #define CODE_ID_MATRIX2OF5 0x0F
- #define CODE_ID_UCCCOMPOS 0x10
- #define CODE_ID_CHINESE2OF5 0x11
- #define CODE_ID_KOREAN3OF5 0x12
- #define CODE_ID_PDF417_ISSNEAN 0x13
- #define CODE_ID_AZTEC_RUNE 0x14
- #define CODE_ID_DATA_MATRIX 0x15
- #define CODE_ID_QRCODE_MICRO 0x16

- #define CODE_ID_MAXICODE 0x17
- #define CODE_ID_US_POSTNET 0x18
- #define CODE_ID_US_PLANET 0x19
- #define CODE_ID_JAPAN_POSTAL 0x1A
- #define CODE_ID_UK_POSTAL 0x1B
- #define CODE_ID_POSTBAR_CA 0x1C
- #define CODE_ID_NETH_KIX 0x1D
- #define CODE_ID_AUS_POST 0x1E
- #define CODE_ID_USPS_4CB 0x1F
- #define CODE_ID_UPU_FICS 0x20
- #define CODE_ID_SCANLET_WEB 0x21
- #define CODE_ID_CUECAT 0x22
- #define AIM_ID_CODE39_32 0x01
- #define AIM_ID_CODE128_ISBT_GS1 0x02
- #define AIM_ID_DATAMATRIX 0x03
- #define AIM_ID_UPC_EAN_COUPON 0x04
- #define AIM_ID_GS1DATABAR 0x05
- #define AIM_ID_CODEBAR 0x06
- #define AIM_ID_CODE93 0x07
- #define AIM_ID_CODE11 0x08
- #define AIM_ID_INTL2OF5 0x09
- #define AIM_ID_PDF417 0x0A
- #define AIM_ID_TLC39 0x0B
- #define AIM_ID_MSI 0x0C
- #define AIM_ID_QRCODE_MICROQR 0x0D
- #define AIM_ID_DISC2OF5 0x0E
- #define AIM_ID_MAXICODE 0x0F
- #define AIM_ID_AZTEC_RUNE 0x10
- #define AIM_ID_X 0x11
- #define AIM_ID_COMP_EC 0x12
- #define AIM_ID_COMP_EE 0x13
- #define AIM_ID_COMP_RS 0x14
- #define SYM_ID_CODE39 0x0001
- #define SYM_ID_CODABAR 0x0002
- #define SYM_ID_CODE128 0x0003
- #define SYM_ID_D25 0x0004
- #define SYM_ID_IATA 0x0005
- #define SYM_ID_ITF 0x0006
- #define SYM_ID_CODE93 0x0007
- #define SYM_ID_UPCA 0x0008
- #define SYM_ID_UPCE 0x0009
- #define SYM_ID_EAN8 0x000A
- #define SYM_ID_EAN13 0x000B
- #define SYM_ID_CODE11 0x000C
- #define SYM_ID_MSI 0x000D
- #define SYM_ID_EAN128 0x000E
- #define SYM_ID_UPCE1 0x000F
- #define SYM_ID_PDF417 0x0010
- #define SYM_ID_CODE39FULL 0x0011
- #define SYM_ID_TRIOPTIC 0x0012
- #define SYM_ID_BOOKLAND 0x0013

- #define SYM_ID_COUPONCODE 0x0014
- #define SYM_ID_ISBT128 0x0015
- #define SYM_ID_MICROPDF 0x0016
- #define SYM_ID_DATAMATRIX 0x0017
- #define SYM_ID_QRCODE 0x0018
- #define SYM_ID_POSTNETUS 0x0019
- #define SYM_ID_PLANETUS 0x001A
- #define SYM_ID_CODE32 0x001B
- #define SYM_ID_ISBT128CONC 0x001C
- #define SYM_ID_POSTALJAPAN 0x001D
- #define SYM_ID_POSTALAUST 0x001E
- #define SYM_ID_POSTALDUTCH 0x001F
- #define SYM_ID_MAXICODE 0x0020
- #define SYM_ID_POSTBARCA 0x0021
- #define SYM_ID_POSTALUK 0x0022
- #define SYM_ID_MACROPDF417 0x0023
- #define SYM_ID_RSS14 0x0024
- #define SYM_ID_RSSLIMIT 0x0025
- #define SYM_ID_RSSEXPAND 0x0026
- #define SYM_ID_SCANLETWEB 0x0027
- #define SYM_ID_CUECAT 0x0028
- #define SYM_ID_UPCA_2 0x0029
- #define SYM_ID_UPCE_2 0x002A
- #define SYM_ID_EAN8_2 0x002B
- #define SYM_ID_EAN13_2 0x002C
- #define SYM_ID_UPCE1_2 0x002D
- #define SYM_ID_CCA_EAN128 0x002E
- #define SYM_ID_CCA_EAN13 0x002F
- #define SYM_ID_CCA_EAN8 0x0030
- #define SYM_ID_CCA_RSSEXPAND 0x0031
- #define SYM_ID_CCA_RSSLIMIT 0x0032
- #define SYM_ID_CCA_RSS14 0x0033
- #define SYM_ID_CCA_UPCA 0x0034
- #define SYM_ID_CCA_UPCE 0x0035
- #define SYM_ID_CCC_EAN128 0x0036
- #define SYM_ID_TLC39 0x0037
- #define SYM_ID_CCB_EAN128 0x0038
- #define SYM_ID_CCB_EAN13 0x0039
- #define SYM_ID_CCB_EAN8 0x003A
- #define SYM_ID_CCB_RSSEXPAND 0x003B
- #define SYM_ID_CCB_RSSLIMIT 0x003C
- #define SYM_ID_CCB_RSS14 0x003D
- #define SYM_ID_CCB_UPCA 0x003E
- #define SYM_ID_CCB_UPCE 0x003F
- #define SYM_ID_KOR3OF5 0x0040
- #define SYM_ID_UPCA_5 0x0041
- #define SYM_ID_UPCE_5 0x0042
- #define SYM_ID_EAN8_5 0x0043
- #define SYM_ID_EAN13_5 0x0044
- #define SYM_ID_UPCE1_5 0x0045
- #define SYM_ID_MACROPDF 0x0046
- #define DISABLE 0x00
- #define ENABLE 0x01

Variables

- const char * symb_AIM [ ]

11.2.1 Macro Definition Documentation

11.2.1.1 #define AIM_ID_AZTEC_RUNE 0x10

11.2.1.2 #define AIM_ID_CODE11 0x08

11.2.1.3 #define AIM_ID_CODE128_ISBT_GS1 0x02

11.2.1.4 #define AIM_ID_CODE39_32 0x01

11.2.1.5 #define AIM_ID_CODE93 0x07

11.2.1.6 #define AIM_ID_CODEBAR 0x06

11.2.1.7 #define AIM_ID_COMP_EC 0x12

11.2.1.8 #define AIM_ID_COMP_EE 0x13

11.2.1.9 #define AIM_ID_COMP_RS 0x14

11.2.1.10 #define AIM_ID_DATAMATRIX 0x03

11.2.1.11 #define AIM_ID_DISC2OF5 0x0E

11.2.1.12 #define AIM_ID_GS1DATABAR 0x05

11.2.1.13 #define AIM_ID_INTL2OF5 0x09

11.2.1.14 #define AIM_ID_MAXICODE 0x0F

11.2.1.15 #define AIM_ID_MSI 0x0C

11.2.1.16 #define AIM_ID_PDF417 0x0A

11.2.1.17 #define AIM_ID_QRCODE_MICROQR 0x0D

11.2.1.18 #define AIM_ID_TLC39 0x0B

11.2.1.19 #define AIM_ID_UPC_EAN_COUPON 0x04

11.2.1.20 #define AIM_ID_X 0x11

11.2.1.21 #define AUTODESCRIMINATE_SUPPLIMENTS 0x02

11.2.1.22 #define CMD_AUTO_BEEP_CONFIG 0x1C

11.2.1.23 #define CMD_BAR_DEV_CLOSE 0x1B

11.2.1.24 #define CMD_BAR_DEV_OPEN 0x1A

11.2.1.25 #define CMD_BARCODE_DATA 0x01

11.2.1.26 #define CMD_BEEP_IMMEDIATE 0x06

11.2.1.27 #define CMD_DISABLE_ALL_SYMB 0x19

11.2.1.28 #define CMD_EN_AIM_PATTERN 0x17

11.2.1.29 #define CMD_EN_CONTINUOUS_RD 0x10

11.2.1.30 #define CMD_EN_MOBILE_PH_MODE 0x12

11.2.1.31 #define CMD_EN_PICKLIST_MODE 0x0C

11.2.1.32 #define CMD_EN_UNIQUE_CODE_REPO 0x11

11.2.1.33 #define CMD_EN_XMIT_FMT 0x16

11.2.1.34 #define CMD_GET_AUTO_BEEP_CONFIG 0x1D

11.2.1.35 #define CMD_MULTI_SCAN 0x0A

11.2.1.36 #define CMD_PASS_THRU 0x03

11.2.1.37 #define CMD_PREFIX_VAL 0x13

11.2.1.38 #define CMD_RESTORE_DEFAULTS 0x0B

11.2.1.39 #define CMD_SCAN_TIMEOUT 0x0D

11.2.1.40 #define CMD_SET_TRIG_MODE 0x05

11.2.1.41 #define CMD_SINGLE_BUTTON_STATUS 0x07

11.2.1.42 #define CMD_SINGLE_GET_APP_VER 0x04

11.2.1.43 #define CMD_SINGLE_GET_DEVICE_ID 0x09

11.2.1.44 #define CMD_SINGLE_GET_FIRM_VER 0x08

11.2.1.45 #define CMD_START_SCAN 0x01

11.2.1.46 #define CMD_STOP_SCAN 0x02

11.2.1.47 #define CMD_SUFFIX1_VAL 0x14

11.2.1.48 #define CMD_SUFFIX2_VAL 0x15

11.2.1.49 #define CMD_SYMBOLOGY 0x18

11.2.1.50 #define CMD_TIMEOUT_BW_DIFF_SYM 0x0F

11.2.1.51 #define CMD_TIMEOUT_BW_SAME_SYM 0x0E

11.2.1.52 #define CODE_ID_AUS_POST 0x1E

11.2.1.53 #define CODE_ID_AZTEC_RUNE 0x14

11.2.1.54 #define CODE_ID_BOOKLAND_EAN 0x0B

11.2.1.55 #define CODE_ID_CHINESE2OF5 0x11

11.2.1.56 #define CODE_ID_CODABAR 0x03

11.2.1.57 #define CODE_ID_CODE11 0x08

11.2.1.58 #define CODE_ID_CODE128_ISBT 0x04

11.2.1.59 #define CODE_ID_CODE39_32 0x02

11.2.1.60 #define CODE_ID_CODE93 0x05

11.2.1.61 #define CODE_ID_COUPONCODE 0x0D

11.2.1.62 #define CODE_ID_CUECAT 0x22

11.2.1.63 #define CODE_ID_DATA_MATRIX 0x15

11.2.1.64 #define CODE_ID_DISC2OF5 0x07

11.2.1.65 #define CODE_ID_GS1DATABAR 0x0E

11.2.1.66 #define CODE_ID_GSI128 0x0A

11.2.1.67 #define CODE_ID_INTL2OF5 0x06

11.2.1.68 #define CODE_ID_JAPAN_POSTAL 0x1A

11.2.1.69 #define CODE_ID_KOREAN3OF5 0x12

11.2.1.70 #define CODE_ID_MATRIX2OF5 0x0F

11.2.1.71 #define CODE_ID_MAXICODE 0x17

11.2.1.72 #define CODE_ID_MSI 0x09

11.2.1.73 #define CODE_ID_NETH_KIX 0x1D

11.2.1.74 #define CODE_ID_PDF417_ISSNEAN 0x13

11.2.1.75 #define CODE_ID_POSTBAR_CA 0x1C

11.2.1.76 #define CODE_ID_QRCODE_MICRO 0x16

11.2.1.77 #define CODE_ID_SCANLET_WEB 0x21

11.2.1.78 #define CODE_ID_TRIOPTIC39 0x0C

11.2.1.79 #define CODE_ID_UCCCOMPOS 0x10

11.2.1.80 #define CODE_ID_UK_POSTAL 0x1B

11.2.1.81 #define CODE_ID_UPC_EA 0x01

11.2.1.82 #define CODE_ID_UPU_FICS 0x20

11.2.1.83 #define CODE_ID_US_PLANET 0x19

11.2.1.84 #define CODE_ID_US_POSTNET 0x18

11.2.1.85 #define CODE_ID_USPS_4CB 0x1F

11.2.1.86 #define DECODE_UPC_EAN_W_SUPPLIMENTS 0x01

11.2.1.87 #define DISABLE 0x00

11.2.1.88 #define ENABLE 0x01

11.2.1.89 #define ENABLE_378_379_SUPPLIMENTS 0x04

11.2.1.90 #define ENABLE_414_419_434_439 0x06

11.2.1.91 #define ENABLE_491_SUPPLIMENTS 0x08

11.2.1.92 #define ENABLE_977_SUPPLIMENT 0x07

11.2.1.93 #define ENABLE_978_979_SUPPLIMENTS 0x05

11.2.1.94 #define ENABLE_SMART_SUPPLIMENT 0x03

11.2.1.95 #define GEN_PID_AIM_PATTERN_EN 0xED

11.2.1.96 #define GEN_PID_AUTO_BEEP_ERROR 0xEA

11.2.1.97 #define GEN_PID_AUTO_BEEP_MODE 0xEC

11.2.1.98 #define GEN_PID_AUTO_BEEP_SCAN 0xEB

11.2.1.99 #define GEN_PID_BEEP_IMMEDIATE 0xFD

11.2.1.100 #define GEN_PID_CONTINUOUS_READ 0xF7

11.2.1.101 #define GEN_PID_MOBILE_PHONE_MODE 0xF5

11.2.1.102 #define GEN_PID_PASS_THRU 0xFF

11.2.1.103 #define GEN_PID_PICKLIST_MODE 0xFB

11.2.1.104 #define GEN_PID_PREFIX_KEY 0xF4

11.2.1.105 #define GEN_PID_PREFIX_VAL 0xF3

11.2.1.106 #define GEN_PID_RESTORE_DEFAULTS 0xFC

11.2.1.107 #define GEN_PID_SCAN_DATA_XMIT_FMT 0xEE

11.2.1.108 #define GEN_PID_SCAN_TIMEOUT 0xFA

11.2.1.109 #define GEN_PID_SET_TRIG_MODE 0xFE

11.2.1.110 #define GEN_PID_SUFFIX1_KEY 0xF2

11.2.1.111 #define GEN_PID_SUFFIX1_VAL 0xF1

11.2.1.112 #define GEN_PID_SUFFIX2_KEY 0xF0

11.2.1.113 #define GEN_PID_SUFFIX2_VAL 0xEF

11.2.1.114 #define GEN_PID_TIMEOUT_BW_DIFF_SYM 0xF8

11.2.1.115 #define GEN_PID_TIMEOUT_BW_SAME_SYM 0xF9

11.2.1.116 #define GEN_PID_UNIQUE_CODE_REPORT 0xF6

11.2.1.117 #define HOST_BUF_SIZE 2048

11.2.1.118 #define IGNORE_UPC_EAN_W_SUPPLIMENTS ∗0x00

11.2.1.119 #define RESP_ACK 0x00000000

11.2.1.120 #define RESP_BARCODE_DATA 0x80000000

11.2.1.121 #define RESP_BUTTON_STATUS 0x80000001

11.2.1.122 #define RESP_NAK 0x00000001

11.2.1.123 #define SMART_SUPPLIMENT_USER_PROG1 0x0B

11.2.1.124 #define SMART_SUPPLIMENT_USER_PROG1_2 0x0C

11.2.1.125 #define SUPPLIMENT_USER_PROG1 0x09

11.2.1.126 #define SUPPLIMENT_USER_PROG1_2 0x0A

11.2.1.127 #define SYM_ID_BOOKLAND 0x0013

11.2.1.128 #define SYM_ID_CCA_EAN128 0x002E

11.2.1.129 #define SYM_ID_CCA_EAN13 0x002F

11.2.1.130 #define SYM_ID_CCA_EAN8 0x0030

11.2.1.131 #define SYM_ID_CCA_RSS14 0x0033

11.2.1.132 #define SYM_ID_CCA_RSSEXPAND 0x0031

11.2.1.133 #define SYM_ID_CCA_RSSLIMIT 0x0032

11.2.1.134 #define SYM_ID_CCA_UPCA 0x0034

11.2.1.135 #define SYM_ID_CCA_UPCE 0x0035

11.2.1.136 #define SYM_ID_CCB_EAN128 0x0038

11.2.1.137 #define SYM_ID_CCB_EAN13 0x0039

11.2.1.138 #define SYM_ID_CCB_EAN8 0x003A

11.2.1.139 #define SYM_ID_CCB_RSS14 0x003D

11.2.1.140 #define SYM_ID_CCB_RSSEXPAND 0x003B

11.2.1.141 #define SYM_ID_CCB_RSSLIMIT 0x003C

11.2.1.142 #define SYM_ID_CCB_UPCA 0x003E

11.2.1.143 #define SYM_ID_CCB_UPCE 0x003F

11.2.1.144 #define SYM_ID_CCC_EAN128 0x0036

11.2.1.145 #define SYM_ID_CODABAR 0x0002

11.2.1.146 #define SYM_ID_CODE11 0x000C

11.2.1.147 #define SYM_ID_CODE128 0x0003

11.2.1.148 #define SYM_ID_CODE32 0x001B

11.2.1.149 #define SYM_ID_CODE39 0x0001

11.2.1.150 #define SYM_ID_CODE39FULL 0x0011

11.2.1.151 #define SYM_ID_CODE93 0x0007

11.2.1.152 #define SYM_ID_COUPONCODE 0x0014

11.2.1.153 #define SYM_ID_CUECAT 0x0028

11.2.1.154 #define SYM_ID_D25 0x0004

11.2.1.155 #define SYM_ID_DATAMATRIX 0x0017

11.2.1.156 #define SYM_ID_EAN128 0x000E

11.2.1.157 #define SYM_ID_EAN13 0x000B

11.2.1.158 #define SYM_ID_EAN13_2 0x002C

11.2.1.159 #define SYM_ID_EAN13_5 0x0044

11.2.1.160 #define SYM_ID_EAN8 0x000A

11.2.1.161 #define SYM_ID_EAN8_2 0x002B

11.2.1.162 #define SYM_ID_EAN8_5 0x0043

11.2.1.163 #define SYM_ID_IATA 0x0005

11.2.1.164 #define SYM_ID_ISBT128 0x0015

11.2.1.165 #define SYM_ID_ISBT128CONC 0x001C

11.2.1.166 #define SYM_ID_ITF 0x0006

11.2.1.167 #define SYM_ID_KOR3OF5 0x0040

11.2.1.168 #define SYM_ID_MACROPDF 0x0046

11.2.1.169 #define SYM_ID_MACROPDF417 0x0023

11.2.1.170 #define SYM_ID_MAXICODE 0x0020

11.2.1.171 #define SYM_ID_MICROPDF 0x0016

11.2.1.172 #define SYM_ID_MSI 0x000D

11.2.1.173 #define SYM_ID_PDF417 0x0010

11.2.1.174 #define SYM_ID_PLANETUS 0x001A

11.2.1.175 #define SYM_ID_POSTALAUST 0x001E

11.2.1.176 #define SYM_ID_POSTALDUTCH 0x001F

11.2.1.177 #define SYM_ID_POSTALJAPAN 0x001D

11.2.1.178 #define SYM_ID_POSTALUK 0x0022

11.2.1.179 #define SYM_ID_POSTBARCA 0x0021

11.2.1.180 #define SYM_ID_POSTNETUS 0x0019

11.2.1.181 #define SYM_ID_QRCODE 0x0018

11.2.1.182 #define SYM_ID_RSS14 0x0024

11.2.1.183 #define SYM_ID_RSSEXPAND 0x0026

11.2.1.184 #define SYM_ID_RSSLIMIT 0x0025

11.2.1.185 #define SYM_ID_SCANLETWEB 0x0027

11.2.1.186 #define SYM_ID_TLC39 0x0037

11.2.1.187 #define SYM_ID_TRIOPTIC 0x0012

11.2.1.188 #define SYM_ID_UPCA 0x0008

11.2.1.189 #define SYM_ID_UPCA_2 0x0029

11.2.1.190 #define SYM_ID_UPCA_5 0x0041

11.2.1.191 #define SYM_ID_UPCE 0x0009

11.2.1.192 #define SYM_ID_UPCE1 0x000F

11.2.1.193 #define SYM_ID_UPCE1_2 0x002D

11.2.1.194 #define SYM_ID_UPCE1_5 0x0045

11.2.1.195 #define SYM_ID_UPCE_2 0x002A

11.2.1.196 #define SYM_ID_UPCE_5 0x0042

11.2.1.197 #define SYM_PID_CHECK_ISBT_TABLE 0x08

11.2.1.198 #define SYM_PID_CODE11_CHK_DIGIT 0x45

11.2.1.199 #define SYM_PID_CODE32_PREFIX 0x24

11.2.1.200 #define SYM_PID_CODE39_BUFFERING 0x2C

11.2.1.201 #define SYM_PID_CODE39_CHK_DIGIT 0x29

11.2.1.202 #define SYM_PID_CODE39_FULL_ASCII 0x2B

11.2.1.203 #define SYM_PID_CONV_CODE39_2_CODE32 0x23

11.2.1.204 #define SYM_PID_CONV_I2OF5_EAN13 0x34

11.2.1.205 #define SYM_PID_CONVERT_UPCE1_2_UPCA 0x1C

11.2.1.206 #define SYM_PID_CONVERT_UPCE_2_UPCA 0x1B

11.2.1.207 #define SYM_PID_COUPON_REPORT 0x1F

11.2.1.208 #define SYM_PID_EAN8_JAN8_EXTEND 0x1D

11.2.1.209 #define SYM_PID_EN_AUST_POST_FMT 0x6D

11.2.1.210 #define SYM_PID_EN_AUSTRALIA_POST 0x6C

11.2.1.211 #define SYM_PID_EN_AZTEC 0x7E

11.2.1.212 #define SYM_PID_EN_AZTEC_INVERSE 0x7F

11.2.1.213 #define SYM_PID_EN_BOOKLAND_EAN 0x0F

11.2.1.214 #define SYM_PID_EN_CHINESE2OF5 0x5C

11.2.1.215 #define SYM_PID_EN_CLSI 0x51

11.2.1.216 #define SYM_PID_EN_CODABAR 0x4C

11.2.1.217 #define SYM_PID_EN_CODE11 0x40

11.2.1.218 #define SYM_PID_EN_CODE128 0x00

11.2.1.219 #define SYM_PID_EN_CODE39 0x21

11.2.1.220 #define SYM_PID_EN_CODE93 0x3B

11.2.1.221 #define SYM_PID_EN_COMP_CC_A_B 0x77

11.2.1.222 #define SYM_PID_EN_COMP_CC_C 0x76

11.2.1.223 #define SYM_PID_EN_COMP_TLC_39 0x78

11.2.1.224 #define SYM_PID_EN_CONV_UPC_EAN 0x75

11.2.1.225 #define SYM_PID_EN_DATA_MATRIX 0x38

11.2.1.226 #define SYM_PID_EN_DATA_MATRIX_INVERSE 0x39

11.2.1.227 #define SYM_PID_EN_DISC2OF5 0x47

11.2.1.228 #define SYM_PID_EN_EAN13_JAN13 0x0E

11.2.1.229 #define SYM_PID_EN_EAN8_JAN8 0x0D

11.2.1.230 #define SYM_PID_EN_GS1128 0x05

11.2.1.231 #define SYM_PID_EN_GS1_128_EMULATION 0x7A

11.2.1.232 #define SYM_PID_EN_GS1_DATABAR 0x71

11.2.1.233 #define SYM_PID_EN_GS1_EXPANDED 0x74

11.2.1.234 #define SYM_PID_EN_GS1_LIMITED 0x72

11.2.1.235 #define SYM_PID_EN_INTER2OF5 0x2D

11.2.1.236 #define SYM_PID_EN_INVERSE1D 0x65

11.2.1.237 #define SYM_PID_EN_ISBN_FORMAT 0x10

11.2.1.238 #define SYM_PID_EN_ISBT 0x06

11.2.1.239 #define SYM_PID_EN_JAPAN_POSTAL 0x6B

11.2.1.240 #define SYM_PID_EN_KOREAN3OF5 0x64

11.2.1.241 #define SYM_PID_EN_MATRIX2OF5 0x5D

11.2.1.242 #define SYM_PID_EN_MAXICODE 0x7D

11.2.1.243 #define SYM_PID_EN_MICRO_PDF417 0x7C

11.2.1.244 #define SYM_PID_EN_MICRO_QR 0x37

11.2.1.245 #define SYM_PID_EN_MIRROR_IMAGES 0x3A

11.2.1.246 #define SYM_PID_EN_MSI 0x54

11.2.1.247 #define SYM_PID_EN_NETHERLANDS_KIX 0x6E

11.2.1.248 #define SYM_PID_EN_NOTIS 0x52

11.2.1.249 #define SYM_PID_EN_PDF417 0x7B

11.2.1.250 #define SYM_PID_EN_QR_CODE 0x35

11.2.1.251 #define SYM_PID_EN_QR_INVERSE 0x36

11.2.1.252 #define SYM_PID_EN_TRIOPTIC_CODE39 0x22

11.2.1.253 #define SYM_PID_EN_UK_POSTAL 0x69

11.2.1.254 #define SYM_PID_EN_UPCA 0x0A

11.2.1.255 #define SYM_PID_EN_UPCE 0x0B

11.2.1.256 #define SYM_PID_EN_UPCE1 0x0C

11.2.1.257 #define SYM_PID_EN_UPU_FICS 0x70

11.2.1.258 #define SYM_PID_EN_US_PLANET 0x67

11.2.1.259 #define SYM_PID_EN_US_POSTNET 0x66

11.2.1.260 #define SYM_PID_EN_USPS_4CB 0x6F

11.2.1.261 #define SYM_PID_I2OF5_CHECK_DIGIT 0x32

11.2.1.262 #define SYM_PID_ISBT_CONCATE 0x07

11.2.1.263 #define SYM_PID_ISBT_CONCATE_REDUN 0x09

11.2.1.264 #define SYM_PID_ISSN_EAN 0x20

11.2.1.265 #define SYM_PID_LTD_SECURITY 0x73

11.2.1.266 #define SYM_PID_M2OF5_CHK_DIGIT 0x62

11.2.1.267 #define SYM_PID_MSI_CHK_DIGIT 0x59

11.2.1.268 #define SYM_PID_MSI_CHK_DIGIT_ALGOR 0x5B

11.2.1.269 #define SYM_PID_SETLEN_1DISCRETE_C11 0x41

11.2.1.270 #define SYM_PID_SETLEN_1DISCRETE_C128 0x01

11.2.1.271 #define SYM_PID_SETLEN_1DISCRETE_C39 0x25

11.2.1.272 #define SYM_PID_SETLEN_1DISCRETE_C93 0x3C

11.2.1.273 #define SYM_PID_SETLEN_1DISCRETE_CBAR 0x4D

11.2.1.274 #define SYM_PID_SETLEN_1DISCRETE_D2OF5 0x48

11.2.1.275 #define SYM_PID_SETLEN_1DISCRETE_I2OF5 0x2E

11.2.1.276 #define SYM_PID_SETLEN_1DISCRETE_M2OF5 0x5E

11.2.1.277 #define SYM_PID_SETLEN_1DISCRETE_MSI 0x55

11.2.1.278 #define SYM_PID_SETLEN_2DISCRETE_C11 0x42

11.2.1.279 #define SYM_PID_SETLEN_2DISCRETE_C128 0x02

11.2.1.280 #define SYM_PID_SETLEN_2DISCRETE_C39 0x26

11.2.1.281 #define SYM_PID_SETLEN_2DISCRETE_C93 0x3D

11.2.1.282 #define SYM_PID_SETLEN_2DISCRETE_CBAR 0x4E

11.2.1.283 #define SYM_PID_SETLEN_2DISCRETE_D2OF5 0x49

11.2.1.284 #define SYM_PID_SETLEN_2DISCRETE_I2OF5 0x2F

11.2.1.285 #define SYM_PID_SETLEN_2DISCRETE_M2OF5 0x5F

11.2.1.286 #define SYM_PID_SETLEN_2DISCRETE_MSI 0x56

11.2.1.287 #define SYM_PID_SETLEN_ANY_C11 0x44

11.2.1.288 #define SYM_PID_SETLEN_ANY_C128 0x04

11.2.1.289 #define SYM_PID_SETLEN_ANY_C39 0x28

11.2.1.290 #define SYM_PID_SETLEN_ANY_C93 0x3F

11.2.1.291 #define SYM_PID_SETLEN_ANY_CBAR 0x50

11.2.1.292 #define SYM_PID_SETLEN_ANY_D2OF5 0x4B

11.2.1.293 #define SYM_PID_SETLEN_ANY_I2OF5 0x31

11.2.1.294 #define SYM_PID_SETLEN_ANY_M2OF5 0x61

11.2.1.295 #define SYM_PID_SETLEN_ANY_MSI 0x58

11.2.1.296 #define SYM_PID_SETLEN_RANGE_C11 0x43

11.2.1.297 #define SYM_PID_SETLEN_RANGE_C128 0x03

11.2.1.298 #define SYM_PID_SETLEN_RANGE_C39 0x27

11.2.1.299 #define SYM_PID_SETLEN_RANGE_C93 0x3E

11.2.1.300 #define SYM_PID_SETLEN_RANGE_CBAR 0x4F

11.2.1.301 #define SYM_PID_SETLEN_RANGE_D2OF5 0x4A

11.2.1.302 #define SYM_PID_SETLEN_RANGE_I2OF5 0x30

11.2.1.303 #define SYM_PID_SETLEN_RANGE_M2OF5 0x60

11.2.1.304 #define SYM_PID_SETLEN_RANGE_MSI 0x57

11.2.1.305 #define SYM_PID_START_STOP_CASE 0x53

11.2.1.306 #define SYM_PID_SUPPLIMENTS_UPC_EAN 0x11

11.2.1.307 #define SYM_PID_UCC_COUPON_EXTEND 0x1E

11.2.1.308 #define SYM_PID_UPC_COMP_MODE 0x79

11.2.1.309 #define SYM_PID_UPC_EAN_JAN_SUPP_REDUN 0x14

11.2.1.310 #define SYM_PID_USER_PROG_SUPP1 0x12

11.2.1.311 #define SYM_PID_USER_PROG_SUPP2 0x13

11.2.1.312 #define SYM_PID_XMIT_CODE11_CHK_DIGIT 0x46

11.2.1.313 #define SYM_PID_XMIT_CODE39_CHK_DIGIT 0x2A

11.2.1.314 #define SYM_PID_XMIT_I2OF5_CHECK_DIGIT 0x33

11.2.1.315 #define SYM_PID_XMIT_M2OF5_CHK_DIGIT 0x63

11.2.1.316 #define SYM_PID_XMIT_MSI_CHK_DIGIT 0x5A

11.2.1.317 #define SYM_PID_XMIT_UK_POST_CHK_DIGIT 0x6A

11.2.1.318 #define SYM_PID_XMIT_UPCA_CHECK_DIGIT 0x15

11.2.1.319 #define SYM_PID_XMIT_UPCA_PREAMBLE 0x18

11.2.1.320 #define SYM_PID_XMIT_UPCE1_CHECK_DIGIT 0x17

11.2.1.321 #define SYM_PID_XMIT_UPCE1_PREAMBLE 0x1A

11.2.1.322 #define SYM_PID_XMIT_UPCE_CHECK_DIGIT 0x16

11.2.1.323 #define SYM_PID_XMIT_UPCE_PREAMBLE 0x19

11.2.1.324 #define SYM_PID_XMIT_US_POST_CHK_DIGIT 0x68

11.2.2 Variable Documentation

11.2.2.1 const char* symb_AIM[]

11.3 decryptedCredentials.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class decryptedCredentials

  Encapsulating data class utilized by VFIConnect.

11.4 GlobalMarkup.d File Reference

11.5 ISCP_HIGH.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class ISCP_HIGH

  Encapsulating data class utilized by VFIBarcode.

11.6 swipeData.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class swipeData

  Encapsulating data class utilized by VFIConnect.

Enumerations

- enum CardType {
  CardTypeNotRecognized, CardTypeVisa, CardTypeMC, CardTypeAmex,
  CardTypeDiscover, CardTypeDCIDisc, CardTypeJCBDisc, CardTypeEnRoute,
  CardTypeNotRecognized, CardTypeVisa, CardTypeMC, CardTypeAmex,
  CardTypeDiscover, CardTypeDCIDisc, CardTypeJCBDisc, CardTypeEnRoute }

11.6.1 Enumeration Type Documentation

11.6.1.1 enum CardType

Enumerator

CardTypeNotRecognized

CardTypeVisa

CardTypeMC

CardTypeAmex

CardTypeDiscover

CardTypeDCIDisc

CardTypeJCBDisc

CardTypeEnRoute

CardTypeNotRecognized

CardTypeVisa

CardTypeMC

CardTypeAmex

CardTypeDiscover

CardTypeDCIDisc

CardTypeJCBDisc

CardTypeEnRoute

11.7 VFIAccessoryMgr.h File Reference

```
#import <Foundation/Foundation.h>
#import <ExternalAccessory/ExternalAccessory.h>
#import "VFIStream.h"
```

Classes

- class VFIAccessoryMgr

Variables

- NSString * DataReceivedNotification_XPI
- NSString * DataReceivedNotification_Barcode
- NSString * DataReceivedNotification_Control
- NSString * DataReceivedNotification_Zontalk
- NSString * DataReceivedNotification_Printer
- NSString * DataReceivedNotification_Sled
- NSString * DataReceivedNotification_XPI_BT
- NSString * DataReceivedNotification_Barcode_BT
- NSString * DataReceivedNotification_Control_BT
- NSString * DataReceivedNotification_Zontalk_BT
- NSString * UpdateEA_XPI
- NSString * UpdateEA_Barcode
- NSString * UpdateEA_Control
- NSString * UpdateEA_Zontalk
- NSString * UpdateEA_Printer
- NSString * UpdateEA_Sled
- NSString * OpenSession_Gen25
- NSString * OpenSession_Printer
- NSString * OpenSession_Sled
- NSString * OpenSession_Gen25_XPI
- NSString * OpenSession_Gen25_Barcode
- NSString * OpenSession_Gen25_Control
- NSString * OpenSession_Gen25_Zontalk
- NSString * OpenSession_Gen20_XPI
- NSString * OpenSession_Gen30_XPI
- NSString * OpenSession_Gen20_Barcode
- NSString * OpenSession_Gen30_Barcode
- NSString * OpenSession_Gen20_Control
- NSString * OpenSession_Gen30_Control

- NSString * OpenSession_Gen20_Zontalk
- NSString * OpenSession_Gen30_Zontalk
- NSString * CloseSession_Gen25
- NSString * CloseSession_Printer
- NSString * CloseSession_Sled
- NSString * CloseSession_Gen20_XPI
- NSString * CloseSession_Gen30_XPI
- NSString * CloseSession_Gen20_Barcode
- NSString * CloseSession_Gen30_Barcode
- NSString * CloseSession_Gen20_Control
- NSString * CloseSession_Gen30_Control
- NSString * CloseSession_Gen20_Zontalk
- NSString * CloseSession_Gen30_Zontalk
- NSString * Connected_Gen25
- NSString * Connected_Printer
- NSString * Connected_Sled
- NSString * Connected_Gen20_XPI
- NSString * Connected_Gen30_XPI
- NSString * Connected_Gen20_Barcode
- NSString * Connected_Gen30_Barcode
- NSString * Connected_Gen20_Control
- NSString * Connected_Gen30_Control
- NSString * Connected_Gen20_Zontalk
- NSString * Connected_Gen30_Zontalk
- NSString * Disconnected_Gen25
- NSString * Disconnected_Printer
- NSString * Disconnected_Sled
- NSString * Disconnected_Gen20_XPI
- NSString * Disconnected_Gen30_XPI
- NSString * Disconnected_Gen20_Barcode
- NSString * Disconnected_Gen30_Barcode
- NSString * Disconnected_Gen20_Control
- NSString * Disconnected_Gen30_Control
- NSString * Disconnected_Gen20_Zontalk
- NSString * Disconnected_Gen30_Zontalk
- NSString * Protocol_Gen25
- NSString * Protocol_Printer
- NSString * Protocol_Sled
- NSString * Protocol_Gen20_XPI
- NSString * Protocol_Gen30_XPI
- NSString * Protocol_Gen20_Barcode
- NSString * Protocol_Gen30_Barcode
- NSString * Protocol_Gen20_Control
- NSString * Protocol_Gen30_Control
- NSString * Protocol_Gen20_Zontalk
- NSString * Protocol_Gen30_Zontalk

11.7.1 Variable Documentation

11.7.1.1 NSString* CloseSession_Gen20_Barcode

11.7.1.2 NSString* CloseSession_Gen20_Control

11.7.1.3 NSString* CloseSession_Gen20_XPI

11.7.1.4 NSString* CloseSession_Gen20_Zontalk

11.7.1.5 NSString* CloseSession_Gen25

11.7.1.6 NSString* CloseSession_Gen30_Barcode

11.7.1.7 NSString* CloseSession_Gen30_Control

11.7.1.8 NSString* CloseSession_Gen30_XPI

11.7.1.9 NSString* CloseSession_Gen30_Zontalk

11.7.1.10 NSString* CloseSession_Printer

11.7.1.11 NSString* CloseSession_Sled

11.7.1.12 NSString* Connected_Gen20_Barcode

11.7.1.13 NSString* Connected_Gen20_Control

11.7.1.14 NSString* Connected_Gen20_XPI

11.7.1.15 NSString* Connected_Gen20_Zontalk

11.7.1.16 NSString* Connected_Gen25

11.7.1.17 NSString* Connected_Gen30_Barcode

11.7.1.18 NSString* Connected_Gen30_Control

11.7.1.19 NSString* Connected_Gen30_XPI

11.7.1.20 NSString* Connected_Gen30_Zontalk

11.7.1.21 NSString* Connected_Printer

11.7.1.22 NSString* Connected_Sled

11.7.1.23 NSString* DataReceivedNotification_Barcode

11.7.1.24 NSString* DataReceivedNotification_Barcode_BT

11.7.1.25 NSString* DataReceivedNotification_Control

11.7.1.26 NSString* DataReceivedNotification_Control_BT

11.7.1.27 NSString* DataReceivedNotification_Printer

11.7.1.28 NSString* DataReceivedNotification_Sled

11.7.1.29 NSString* DataReceivedNotification_XPI

11.7.1.30 NSString* DataReceivedNotification_XPI_BT

11.7.1.31 NSString* DataReceivedNotification_Zontalk

11.7.1.32 NSString* DataReceivedNotification_Zontalk_BT

11.7.1.33 NSString* Disconnected_Gen20_Barcode

11.7.1.34 NSString* Disconnected_Gen20_Control

11.7.1.35 NSString* Disconnected_Gen20_XPI

11.7.1.36 NSString* Disconnected_Gen20_Zontalk

11.7.1.37 NSString* Disconnected_Gen25

11.7.1.38 NSString* Disconnected_Gen30_Barcode

11.7.1.39 NSString* Disconnected_Gen30_Control

11.7.1.40 NSString* Disconnected_Gen30_XPI

11.7.1.41 NSString* Disconnected_Gen30_Zontalk

11.7.1.42 NSString* Disconnected_Printer

11.7.1.43 NSString* Disconnected_Sled

11.7.1.44 NSString* OpenSession_Gen20_Barcode

11.7.1.45 NSString* OpenSession_Gen20_Control

11.7.1.46 NSString* OpenSession_Gen20_XPI

11.7.1.47 NSString* OpenSession_Gen20_Zontalk

11.7.1.48 NSString* OpenSession_Gen25

11.7.1.49 NSString* OpenSession_Gen25_Barcode

11.7.1.50 NSString* OpenSession_Gen25_Control

11.7.1.51 NSString* OpenSession_Gen25_XPI

11.7.1.52 NSString* OpenSession_Gen25_Zontalk

11.7.1.53 NSString* OpenSession_Gen30_Barcode

11.7.1.54 NSString* OpenSession_Gen30_Control

11.7.1.55 NSString* OpenSession_Gen30_XPI

11.7.1.56 NSString* OpenSession_Gen30_Zontalk

11.7.1.57 NSString* OpenSession_Printer

11.7.1.58 NSString* OpenSession_Sled

11.7.1.59 NSString* Protocol_Gen20_Barcode

11.7.1.60 NSString* Protocol_Gen20_Control

11.7.1.61 NSString* Protocol_Gen20_XPI

11.7.1.62 NSString* Protocol_Gen20_Zontalk

11.7.1.63 NSString* Protocol_Gen25

11.7.1.64 NSString* Protocol_Gen30_Barcode

11.7.1.65 NSString* Protocol_Gen30_Control

11.7.1.66 NSString* Protocol_Gen30_XPI

11.7.1.67 NSString* Protocol_Gen30_Zontalk

11.7.1.68 NSString* Protocol_Printer

11.7.1.69 NSString* Protocol_Sled

11.7.1.70 NSString* UpdateEA_Barcode

11.7.1.71 NSString* UpdateEA_Control

11.7.1.72 NSString* UpdateEA_Printer

11.7.1.73 NSString* UpdateEA_Sled

11.7.1.74 NSString* UpdateEA_XPI

11.7.1.75 NSString* UpdateEA_Zontalk

11.8 VFIBarcode.h File Reference

```
#import <Foundation/Foundation.h>
#import <ExternalAccessory/ExternalAccessory.h>
#import "ISCP_HIGH.h"
#import <UIKit/UIKit.h>
#import "VFIAccessoryMgr.h"
#import "VFIStream.h"
#import "Barcode_host.h"
#import "VFIEncryptedBarcode.h"
```

Classes

- protocol <VFIBarcodeDelegate>

  Protocol methods established for VFIBarcode class.
- class VFIBarcode

  API methods for e210/e255/e315 Pinpad Barcode Scanner.

Enumerations

- enum TYPE {
  TYPE_SETUP_READ =0x40, TYPE_SETUP_WRITE =0x41, TYPE_CONTROL =0x42, TYPE_STATUS_READ =0x43,
  TYPE_PERMISSION_READ =0x44, TYPE_PERMISSION_WRITE =0x45, TYPE_SETUP_REPLY =0x50, TYPE_RESULT =0x51,
  TYPE_STATUS_REPLY =0x53, TYPE_SETUP_PERMISSION_REPLY =0x54, TYPE_BARCODE_DATA =0x60, TYPE_EVENT_NOTIFICATION =0x61,
  TYPE_SETUP_BARCODE_DATA =0x62 }

  The frame type indicates what kind of frame is being sent or received.
- enum barcodeTypes {
  BC_EAN_UPC2 ='E', BC_EAN_UPC1 ='X', BC_CODABAR ='F', BC_CODABLOCK ='O',
  BC_CODE11 ='H', BC_CODE39 ='A', BC_CODE93 ='G', BC_CODE128 ='C',
  BC_DATAMATRIX ='d', BC_GS1 ='e', BC_INTERLEAVED_2_OF_5 ='I', BC_MAXICODE ='U',
  BC_MSICODE ='M', BC_PDF417 ='L', BC_PLESSYCODE ='P', BC_QRCODE ='Q',
  BC_STAN_2_OF_5_2BARS ='R', BC_STAN_2_OF_5_3BARS ='S', BC_TELEPEN ='B' }

  Barcode types.
- enum scanningModes { BCS_BUTTON_EDGE = 0, BCS_BUTTON_LEVEL, BCS_BUTTON_SOFT, BCS_BUTTON_PASSIVE }

  Scanning Modes (ISCP)
- enum commResult {
  COMM_RESULT_ACK =0, COMM_RESULT_NAK, COMM_RESULT_BUSY, COMM_RESULT_BAD_RESPONSE,
  COMM_RESULT_NO_RESPONSE }

  Verix BCS Application Host Responses (ISCP)
- enum triggerResponse { BCS_TRIGGER_RELEASED = 0, BCS_TRIGGER_LEFT_PRESSED, BCS_TRIGGER_RIGHT_PRESSED, BCS_TRIGGER_BOTH_PRESSED }

  BCS Application Trigger Responses (ISCP)

11.8.1 Enumeration Type Documentation

11.8.1.1 enum barcodeTypes

Barcode types.

(ISCP)

Values returned when option to identify returning barcode types is ON when barcodeTypeEnabled: (VFIBarcode) is passed `TRUE`.

Enumerator

BC_EAN_UPC2

BC_EAN_UPC1

BC_CODABAR

BC_CODABLOCK

BC_CODE11

BC_CODE39

BC_CODE93

BC_CODE128

BC_DATAMATRIX

BC_GS1

BC_INTERLEAVED_2_OF_5

BC_MAXICODE

BC_MSICODE

BC_PDF417

BC_PLESSYCODE

BC_QRCODE

BC_STAN_2_OF_5_2BARS

BC_STAN_2_OF_5_3BARS

BC_TELEPEN

11.8.1.2 enum commResult

Verix BCS Application Host Responses (ISCP)

Enumerator

COMM_RESULT_ACK Response Acknowledged.

COMM_RESULT_NAK Response Not Acknowledged.

COMM_RESULT_BUSY Verix Application Busy.

COMM_RESULT_BAD_RESPONSE Verix Application Bad Response.

COMM_RESULT_NO_RESPONSE Verix Application No Response.

11.8.1.3 enum scanningModes

Scanning Modes (ISCP)

Enumerator

BCS_BUTTON_EDGE Scanning session begin on trigger press, and continues after trigger released. Scanning completes upon successful scan capture, abortScan() or timeout.

BCS_BUTTON_LEVEL Scanning session begin on trigger press. Scanning completes upon successful scan capture, abortyScan(), trigger release or timeout.

BCS_BUTTON_SOFT Triggers send events to framework for integrator to control scanning session via software calls.

BCS_BUTTON_PASSIVE Triggers do not send events to framework. Integrator can control scanning session via software calls.

11.8.1.4 enum triggerResponse

BCS Application Trigger Responses (ISCP)

Enumerator

BCS_TRIGGER_RELEASED Trigger Released.

BCS_TRIGGER_LEFT_PRESSED Left Trigger Pressed.

BCS_TRIGGER_RIGHT_PRESSED Right Trigger Pressed.

BCS_TRIGGER_BOTH_PRESSED Both Triggers Pressed.

11.8.1.5 enum TYPE

The frame type indicates what kind of frame is being sent or received.

(ISCP)

Enumerator

TYPE_SETUP_READ The host sends a Setup Read to know how the scanner is configured.

TYPE_SETUP_WRITE The host sends a Setup Write to configure the scanner or modify the current configuration.

TYPE_CONTROL Control Command is used to control the scanner.

TYPE_STATUS_READ Status Read is used to find out information on the status of certain parameters in the scanner.

TYPE_PERMISSION_READ Setup Permission Read frame is used to find out what setup parameters can or cannot be changed by reading configuration bar codes.

TYPE_PERMISSION_WRITE Setup Permissions allow the host to prevent users from modifying the scanner's setup parameters when reading configuration bar codes.

TYPE_SETUP_REPLY A Setup Reply is used to send to the host information about the configuration of the scanner. This frame is sent in response to a Setup Read.

TYPE_RESULT All Result frames, except Unknown Frame Type, are sent in response to three different kinds of frames: Setup Write, Control Command and Setup Permission Write.

TYPE_STATUS_REPLY After receiving a Status Read from the host, the scanner replies by sending the requested status parameter values to the host.

TYPE_SETUP_PERMISSION_REPLY After receiving a Setup Permission Read from the host, the scanner replies by sending the permission status of the requested setup parameters.

TYPE_BARCODE_DATA A Barcode Data frame is a scanner-initiated frame. When the scanner reads a bar code, data is sent from the scanner to the host.

TYPE_EVENT_NOTIFICATION An Event notification is a scanner-initiated frame. It is used to inform the host when certain events have taken place such as an unsuccessful decoding, etc.

TYPE_SETUP_BARCODE_DATA This frame is used to send configuration barcode data via the scanner to the host in packet format.

11.9 VFIBTBridge.h File Reference

```
#import <Foundation/Foundation.h>
#import <ExternalAccessory/ExternalAccessory.h>
```

Classes

- class VFIBTBridge

11.10 VFICardData.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFICardData

  Encapsulating data class utilized by VFIPinpad.

11.11 VFICipheredData.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFICipheredData

  Encapsulating data class utilized by VFIPinpad.

11.12 VFIControl.h File Reference

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
#import <ExternalAccessory/ExternalAccessory.h>
#import "VFISoftwareVersion.h"
#import "VFIKeypadVersion.h"
#import "VFIAccessoryMgr.h"
#import "VFIStream.h"
#import "VFIGen3DeviceInfo.h"
```

Classes

- protocol <VFIControlDelegate>

  Protocol methods established for VFIControl class.
- struct POWER_STATUS

  POWER_STATUS structure.
- class VFIControl

  API methods for e210/e255/e315 Pinpad Control.

11.13 VFIDataEntry.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIDataEntry

  Encapsulating data class utilized by VFIPinpad.

11.14 VFIDecryptedCredentials.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class decryptedCredentials

  Encapsulating data class utilized by VFIConnect.

11.15 VFIDiagnostics.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIDiagnostics

  Encapsulating data class utilized by VFIPinpad.

11.16 VFIEMAFiles.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEMAFiles

  Encapsulating data class utilized by VFIPinpad.

11.17 VFIEMVAuthorization.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEMVAuthorization

  Encapsulating data class utilized by VFIPinpad.

11.18 VFIEMVCompletionData.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEMVCompletionData

  Encapsulating data class utilized by VFIPinpad.

11.19 VFIEMVConfiguration.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEMVConfiguration

  Encapsulating data class utilized by VFIPinpad.

11.20 VFIEMVResponse.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEMVResponse

  Encapsulating data class utilized by VFIPinpad.

11.21 VFIEMVTags.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEMVTags

  Encapsulating data class utilized by VFIPinpad.

11.22 VFIEncryptedBarcode.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEncryptedBarcode

  Encapsulating data class representing Encrypted Barcode Data.

11.23 VFIEncryptionData.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIEncryptionData

  Encapsulating data class utilized by VFIPinpad.

11.24 VFIFinancialData.h File Reference

```
#import <Foundation/Foundation.h>
#import "VFICardData.h"
#import "VFIEncryptionData.h"
#import "VFIEMVAuthorization.h"
#import "VFIDataEntry.h"
#import "VFIEMVTags.h"
```

Classes

- class VFIFinancialData

  Encapsulating data class utilized by VFIPinpad.

11.25 VFIGen3DeviceInfo.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIGen3DeviceInfo

  Encapsulating data class utilized by VFIPinpad.

11.26 VFIKeypadVersion.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFIKeypadVersion

  Encapsulating data class utilized by VFIControl.

11.27 VFIPinpad.h File Reference

```
#import <UIKit/UIKit.h>
```

```
#import <Foundation/Foundation.h>
#import <ExternalAccessory/ExternalAccessory.h>
#import "VFICardData.h"
#import "VFIEMAFiles.h"
#import "VFIBINRange.h"
#import "VFIDataEntry.h"
#import "VFICipheredData.h"
#import "VFIZontalk.h"
#import "VFIDiagnostics.h"
#import "VFIEMVAuthorization.h"
#import "VFIEMVCompletionData.h"
#import "VFIEMVConfiguration.h"
#import "VFIEMVResponse.h"
#import "VFIEncryptionData.h"
#import "VFIFinancialData.h"
#import "VFIEMVTags.h"
#import "VFIZipArchive.h"
#import "VFIControl.h"
#import "VFIBarcode.h"
#import "AsyncSocket.h"
#import "NSData-AES.h"
#import "Base64.h"
#include "VFI_unzip.h"
```

Classes

- protocol <VFIPinpadDelegate>

  Protocol methods established for VFIPinpad class NEW.
- class VFIPinpad

  API methods for e210/e255/e315 for XPI.

Enumerations

- enum VFIErrorCode { VFIResultCode_Framework_Timeout = -1, VFIResultCode_ACK_Timeout = -2, VFIResultCode_Framework_Invalid_Data = -3, VFIResultCode_Framework_No_Blocking = -4, VFIResultCode_Device_Not_Available = -5 }

  VMF Generated Error Code Responses.
- enum EncryptionMode { EncryptionMode_VSP =1, EncryptionMode_PKI =2, EncryptionMode_NOE =0, EncryptionMode_VSD =5 }

  XPI Encryption Modes.
- enum SEVERITY_LOG_LEVELS { SEV_DEBUG =0, SEV_INFO_PINPAD =1, SEV_WARN =2, SEV_ERROR =3, SEV_FATAL =4, SEV_UNKOWN =5 }

  Logging Severity Levels.

11.27.1 Enumeration Type Documentation

11.27.1.1 enum EncryptionMode

XPI Encryption Modes.

Enumerator

EncryptionMode_VSP

EncryptionMode_PKI

EncryptionMode_NOE

EncryptionMode_VSD

11.27.1.2 enum SEVERITY_LOG_LEVELS

Logging Severity Levels.

Enumerator

SEV_DEBUG

SEV_INFO_PINPAD

SEV_WARN

SEV_ERROR

SEV_FATAL

SEV_UNKOWN

11.27.1.3 enum VFIErrorCode

VMF Generated Error Code Responses.

Enumerator

VFIResultCode_Framework_Timeout Command never received reply to command sent. XPI is unresponsive.

VFIResultCode_ACK_Timeout Command never received <ACK> of command sent. XPI is unresponsive.

VFIResultCode_Framework_Invalid_Data Data passed to method is out of bounds or invalid.

VFIResultCode_Framework_No_Blocking value returned on when non-blocking is enabled

VFIResultCode_Device_Not_Available value returned when command was sent when device was offline

11.28 VFISoftwareVersion.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class VFISoftwareVersion

  Encapsulating data class utilized by VFIControl.

11.29 VFIStream.h File Reference

```
#import <Foundation/Foundation.h>
#import <ExternalAccessory/ExternalAccessory.h>
```

Classes

- protocol <VFIStreamDelegate>
- class VFIStream

11.30 VFISwipeData.h File Reference

```
#import <Foundation/Foundation.h>
```

Classes

- class swipeData

  Encapsulating data class utilized by VFIConnect.

Enumerations

- enum CardType {
  CardTypeNotRecognized, CardTypeVisa, CardTypeMC, CardTypeAmex, CardTypeDiscover, CardTypeDCIDisc, CardTypeJCBDisc, CardTypeEnRoute, CardTypeNotRecognized, CardTypeVisa, CardTypeMC, CardTypeAmex, CardTypeDiscover, CardTypeDCIDisc, CardTypeJCBDisc, CardTypeEnRoute }

11.30.1 Enumeration Type Documentation

11.30.1.1 enum CardType

Enumerator

CardTypeNotRecognized

CardTypeVisa

CardTypeMC

CardTypeAmex

CardTypeDiscover

CardTypeDCIDisc

CardTypeJCBDisc

CardTypeEnRoute

CardTypeNotRecognized

CardTypeVisa

CardTypeMC

CardTypeAmex

CardTypeDiscover

CardTypeDCIDisc

CardTypeJCBDisc

CardTypeEnRoute

Index

<AsyncSocketDelegate>, 142
<VFIBarcodeDelegate>, 183
<VFIControlDelegate>, 212
<VFIPinpadDelegate>, 362
<VFIStreamDelegate>, 372
_accessory
    VFIStream, 372
_isBTDevice
    VFIAccessoryMgr, 156
_protocolString
    VFIStream, 372
_readData
    VFIStream, 372
_session
    VFIStream, 372
_sessionOpen
    VFIStream, 372
_sessionOpen_Barcode
    VFIStream, 372
_sessionOpen_Control
    VFIStream, 372
_sessionOpen_XPI
    VFIStream, 372
_sessionOpen_Zontalk
    VFIStream, 372
_sessions
    VFIAccessoryMgr, 156
_unusedProtocols
    VFIAccessoryMgr, 156
_writeData
    VFIStream, 372

AID
    VFICardData, 191
AIM_ID_AZTEC_RUNE
    Barcode_host.h, 383
AIM_ID_CODE11
    Barcode_host.h, 383
AIM_ID_CODE39_32
    Barcode_host.h, 383
AIM_ID_CODE93
    Barcode_host.h, 383
AIM_ID_CODEBAR
    Barcode_host.h, 383
AIM_ID_COMP_EC
    Barcode_host.h, 383
AIM_ID_COMP_EE
    Barcode_host.h, 383
AIM_ID_COMP_RS
    Barcode_host.h, 383
AIM_ID_DATAMATRIX
    Barcode_host.h, 383
AIM_ID_DISC2OF5
    Barcode_host.h, 383
AIM_ID_GS1DATABAR
    Barcode_host.h, 383
AIM_ID_INTL2OF5
    Barcode_host.h, 383
AIM_ID_MAXICODE
    Barcode_host.h, 383
AIM_ID_MSI
    Barcode_host.h, 383
AIM_ID_PDF417
    Barcode_host.h, 383
AIM_ID_TLC39
    Barcode_host.h, 383
AIM_ID_X
    Barcode_host.h, 383
AIMID
    symb_INFO, 154
ALLOW_MESSAGES
    VFIEMAFiles, 222
AMMVersion
    VFIGen3DeviceInfo, 243
abortScan
    VFIBarcode, 162
acceptEncryptPIN:
    VFIPinpad, 258
acceptOnInterface:port:error:
    AsyncSocket, 134
acceptOnPort:error:
    AsyncSocket, 134
accessory
    VFIStream, 372
accountEntryTimeout
    VFIPinpad, 358
accountNumber
    VFICardData, 191
accountSelected
    VFIDataEntry, 218
accountSelection

VFIEncryptionData, 240
ackTimeout
VFIPinpad, 358
acquireID
VFIEMVConfiguration, 229
activateBarcodeType:
VFIBarcode, 162
addRunLoopMode:
AsyncSocket, 134
additionalCapacity
VFIEMVConfiguration, 229
additionalPromptResponses
VFIFinancialData, 241
additionalPrompts
VFIFinancialData, 241
amountConfirmed
VFIDataEntry, 218
amountEntry:message1:message2:
VFIPinpad, 259
amountIntroduction:amountFormat:message1:message2:
VFIPinpad, 260
amountResponse
VFIPinpad, 359
AppBuild
VFISoftwareVersion, 369
appLabel
VFICardData, 191
AppMajor
VFISoftwareVersion, 369
AppMinor
VFISoftwareVersion, 369
appPreferredName
VFICardData, 191
AsyncSocket.h
AsyncSocketCFSocketError, 376
AsyncSocketCanceledError, 376
AsyncSocketConnectTimeoutError, 376
AsyncSocketNoError, 376
AsyncSocketReadMaxedOutError, 376
AsyncSocketReadTimeoutError, 376
AsyncSocketWriteTimeoutError, 376
AsyncSocketCFSocketError
AsyncSocket.h, 376
AsyncSocketCanceledError
AsyncSocket.h, 376
AsyncSocketConnectTimeoutError
AsyncSocket.h, 376
AsyncSocketNoError
AsyncSocket.h, 376
AsyncSocketReadMaxedOutError
AsyncSocket.h, 376
AsyncSocketReadTimeoutError
AsyncSocket.h, 376
AsyncSocketWriteTimeoutError
AsyncSocket.h, 376
AsyncSocket, 131
acceptOnInterface:port:error:, 134
acceptOnPort:error:, 134
addRunLoopMode:, 134
CRData, 135
CRLFData, 135
canSafelySetDelegate, 134
connectToAddress:error:, 134
connectToAddress:viaInterfaceAddress:withTimeout-:error:, 134
connectToAddress:withTimeout:error:, 134
connectToHost:onPort:error:, 135
connectToHost:onPort:withTimeout:error:, 135
connectedAddress, 134
connectedHost, 134
connectedPort, 134
delegate, 135
description, 135
disconnect, 135
disconnectAfterReading, 135
disconnectAfterReadingAndWriting, 135
disconnectAfterWriting, 135
enablePreBuffering, 136
getCFReadStream, 136
getCFSocket, 136
getCFWriteStream, 136
init, 136
initWithDelegate:, 136
initWithDelegate:userData:, 136
isConnected, 136
isIPv4, 136
isIPv6, 136
LFData, 136
localAddress, 136
localHost, 136
localPort, 136
moveToRunLoop:, 136
partialReadBuffer, 141
progressOfReadReturningTag:bytesDone:total:, 137
progressOfWriteReturningTag:bytesDone:total:, 137
readDataToData:withTimeout:buffer:bufferOffset-:maxLength:tag:, 137
readDataToData:withTimeout:buffer:bufferOffset:tag:, 137
readDataToData:withTimeout:maxLength:tag:, 138
readDataToData:withTimeout:tag:, 138
readDataToLength:withTimeout:buffer:bufferOffset-:tag:, 138
readDataToLength:withTimeout:tag:, 139
readDataWithTimeout:buffer:bufferOffset:max-Length:tag:, 139
readDataWithTimeout:buffer:bufferOffset:tag:, 139
readDataWithTimeout:tag:, 139

removeRunLoopMode:, 139
runLoopModes, 139
setDelegate:, 140
setRunLoopModes:, 140
setUserData:, 140
startTLS:, 140
theConnectTimer, 141
theContext, 141
theCurrentRead, 141
theCurrentWrite, 141
theDelegate, 141
theFlags, 141
theNativeSocket4, 141
theNativeSocket6, 141
theReadQueue, 141
theReadStream, 141
theReadTimer, 141
theRunLoop, 141
theRunLoopModes, 142
theSocket4, 142
theSocket6, 142
theSource4, 142
theSource6, 142
theUserData, 142
theWriteQueue, 142
theWriteStream, 142
theWriteTimer, 142
unreadData, 141
userData, 141
write:, 141
writeData:withTimeout:tag:, 141
ZeroData, 141
AsyncSocket.h, 375
AsyncSocketError, 375
AsyncSocketErrorDomain, 376
AsyncSocketException, 376
AsyncSocketDelegate-p
onSocket:didAcceptNewSocket:, 143
onSocket:didConnectToHost:port:, 143
onSocket:didReadData:withTag:, 143
onSocket:didReadPartialDataOfLength:tag:, 143
onSocket:didWriteDataWithTag:, 143
onSocket:didWritePartialDataOfLength:tag:, 143
onSocket:shouldTimeoutReadWithTag:elapsed-:bytesDone:, 143
onSocket:shouldTimeoutWriteWithTag:elapsed-:bytesDone:, 144
onSocket:wantsRunLoopForNewSocket:, 144
onSocket:willDisconnectWithError:, 144
onSocketDidDisconnect:, 144
onSocketDidSecure:, 144
onSocketWillConnect:, 145
AsyncSocketError
AsyncSocket.h, 375
AsyncSocketErrorDomain
AsyncSocket.h, 376
AsyncSocketException
AsyncSocket.h, 376
autoSelectApplication
VFIEMVConfiguration, 229

b1
T_BEEP_DEF, 155
b2
T_BEEP_DEF, 155
BC_CODABAR
VFIBarcode.h, 403
BC_CODABLOCK
VFIBarcode.h, 403
BC_CODE11
VFIBarcode.h, 403
BC_CODE128
VFIBarcode.h, 403
BC_CODE39
VFIBarcode.h, 403
BC_CODE93
VFIBarcode.h, 403
BC_DATAMATRIX
VFIBarcode.h, 403
BC_EAN_UPC1
VFIBarcode.h, 403
BC_EAN_UPC2
VFIBarcode.h, 403
BC_GS1
VFIBarcode.h, 403
BC_INTERLEAVED_2_OF_5
VFIBarcode.h, 403
BC_MAXICODE
VFIBarcode.h, 403
BC_MSICODE
VFIBarcode.h, 403
BC_PDF417
VFIBarcode.h, 403
BC_PLESSYCODE
VFIBarcode.h, 403
BC_QRCODE
VFIBarcode.h, 403
BC_STAN_2_OF_5_2BARS
VFIBarcode.h, 403
BC_STAN_2_OF_5_3BARS
VFIBarcode.h, 403
BC_TELEPEN
VFIBarcode.h, 403
BCS_BUTTON_EDGE
VFIBarcode.h, 404
BCS_BUTTON_LEVEL
VFIBarcode.h, 404
BCS_BUTTON_PASSIVE

VFIBarcode.h, 404
BCS_BUTTON_SOFT
VFIBarcode.h, 404
BCS_TRIGGER_BOTH_PRESSED
VFIBarcode.h, 404
BCS_TRIGGER_LEFT_PRESSED
VFIBarcode.h, 404
BCS_TRIGGER_RELEASED
VFIBarcode.h, 404
BCS_TRIGGER_RIGHT_PRESSED
VFIBarcode.h, 404
bPause
T_BEEP_DEF, 155
BTconnected
VFIBarcode, 182
VFIControl, 211
VFIPinpad, 359
Barcode_host.h, 376
AIM_ID_CODE11, 383
AIM_ID_CODE39_32, 383
AIM_ID_CODE93, 383
AIM_ID_CODEBAR, 383
AIM_ID_COMP_EC, 383
AIM_ID_COMP_EE, 383
AIM_ID_COMP_RS, 383
AIM_ID_DATAMATRIX, 383
AIM_ID_DISC2OF5, 383
AIM_ID_GS1DATABAR, 383
AIM_ID_INTL2OF5, 383
AIM_ID_MAXICODE, 383
AIM_ID_MSI, 383
AIM_ID_PDF417, 383
AIM_ID_TLC39, 383
AIM_ID_X, 383
CMD_BAR_DEV_OPEN, 383
CMD_BARCODE_DATA, 384
CMD_EN_XMIT_FMT, 384
CMD_MULTI_SCAN, 384
CMD_PASS_THRU, 384
CMD_PREFIX_VAL, 384
CMD_SCAN_TIMEOUT, 384
CMD_START_SCAN, 384
CMD_STOP_SCAN, 384
CMD_SUFFIX1_VAL, 384
CMD_SUFFIX2_VAL, 384
CMD_SYMBOLOGY, 384
CODE_ID_AUS_POST, 385
CODE_ID_CHINESE2OF5, 385
CODE_ID_CODABAR, 385
CODE_ID_CODE11, 385
CODE_ID_CODE39_32, 385
CODE_ID_CODE93, 385
CODE_ID_CUECAT, 385
CODE_ID_DISC2OF5, 385
CODE_ID_GS1DATABAR, 385
CODE_ID_GSI128, 385
CODE_ID_INTL2OF5, 385
CODE_ID_KOREAN3OF5, 385
CODE_ID_MATRIX2OF5, 385
CODE_ID_MAXICODE, 385
CODE_ID_MSI, 385
CODE_ID_NETH_KIX, 385
CODE_ID_TRIOPTIC39, 386
CODE_ID_UCCCOMPOS, 386
CODE_ID_UPC_EA, 386
CODE_ID_UPU_FICS, 386
CODE_ID_USPS_4CB, 386
DISABLE, 386
ENABLE, 386
ENABLE_414_419_434_439, 386
HOST_BUF_SIZE, 387
RESP_ACK, 387
RESP_BARCODE_DATA, 387
RESP_NAK, 387
SYM_ID_BOOKLAND, 387
SYM_ID_CCA_EAN128, 387
SYM_ID_CCA_EAN13, 388
SYM_ID_CCA_EAN8, 388
SYM_ID_CCA_RSS14, 388
SYM_ID_CCA_UPCA, 388
SYM_ID_CCA_UPCE, 388
SYM_ID_CCB_EAN128, 388
SYM_ID_CCB_EAN13, 388
SYM_ID_CCB_EAN8, 388
SYM_ID_CCB_RSS14, 388
SYM_ID_CCB_UPCA, 388
SYM_ID_CCB_UPCE, 388
SYM_ID_CCC_EAN128, 388
SYM_ID_CODABAR, 388
SYM_ID_CODE11, 388
SYM_ID_CODE128, 388
SYM_ID_CODE32, 388
SYM_ID_CODE39, 388
SYM_ID_CODE39FULL, 388
SYM_ID_CODE93, 388
SYM_ID_COUPONCODE, 388
SYM_ID_CUECAT, 388
SYM_ID_D25, 388
SYM_ID_DATAMATRIX, 389
SYM_ID_EAN128, 389
SYM_ID_EAN13, 389
SYM_ID_EAN13_2, 389
SYM_ID_EAN13_5, 389
SYM_ID_EAN8, 389
SYM_ID_EAN8_2, 389
SYM_ID_EAN8_5, 389
SYM_ID_IATA, 389
SYM_ID_ISBT128, 389

SYM_ID_ISBT128CONC, 389
SYM_ID_ITF, 389
SYM_ID_KOR3OF5, 389
SYM_ID_MACROPDF, 389
SYM_ID_MACROPDF417, 389
SYM_ID_MAXICODE, 389
SYM_ID_MICROPDF, 389
SYM_ID_MSI, 389
SYM_ID_PDF417, 389
SYM_ID_PLANETUS, 389
SYM_ID_POSTALAUST, 389
SYM_ID_POSTALUK, 389
SYM_ID_POSTBARCA, 389
SYM_ID_POSTNETUS, 389
SYM_ID_QRCODE, 390
SYM_ID_RSS14, 390
SYM_ID_RSSEXPAND, 390
SYM_ID_RSSLIMIT, 390
SYM_ID_SCANLETWEB, 390
SYM_ID_TLC39, 390
SYM_ID_TRIOPTIC, 390
SYM_ID_UPCA, 390
SYM_ID_UPCA_2, 390
SYM_ID_UPCA_5, 390
SYM_ID_UPCE, 390
SYM_ID_UPCE1, 390
SYM_ID_UPCE1_2, 390
SYM_ID_UPCE1_5, 390
SYM_ID_UPCE_2, 390
SYM_ID_UPCE_5, 390
SYM_PID_EN_AZTEC, 391
SYM_PID_EN_CLSI, 391
SYM_PID_EN_CODE11, 391
SYM_PID_EN_CODE128, 391
SYM_PID_EN_CODE39, 391
SYM_PID_EN_CODE93, 391
SYM_PID_EN_GS1128, 391
SYM_PID_EN_ISBT, 392
SYM_PID_EN_MSI, 392
SYM_PID_EN_NOTIS, 392
SYM_PID_EN_PDF417, 392
SYM_PID_EN_UPCA, 392
SYM_PID_EN_UPCE, 392
SYM_PID_EN_UPCE1, 392
SYM_PID_ISSN_EAN, 393
symb_AIM, 395
barcodeConnected
VFIBarcode, 181
barcodeConnected:
VFIBarcodeDelegate-p, 185
barcodeDataReceived:
VFIBarcodeDelegate-p, 186
barcodeDataSent:
VFIBarcodeDelegate-p, 186
barcodeFirmwareRevision
VFIBarcode, 182
barcodeHardwareRevision
VFIBarcode, 182
barcodeInitialized:
VFIBarcodeDelegate-p, 186
barcodeLogEntry:withSeverity:
VFIBarcodeDelegate-p, 186
barcodeManufacturer
VFIBarcode, 182
barcodeMessage:
VFIBarcodeDelegate-p, 186
barcodeModelNumber
VFIBarcode, 182
barcodeName
VFIBarcode, 182
barcodeNotfication:
VFIAccessoryMgr, 156
barcodeReconnectFinished
VFIBarcodeDelegate-p, 187
barcodeReconnectStarted
VFIBarcodeDelegate-p, 187
barcodeScanData:
VFIBarcodeDelegate-p, 187
barcodeScanData:barcodeType:
VFIBarcodeDelegate-p, 187
barcodeScanData:code:symbol:aim:
VFIBarcodeDelegate-p, 187
barcodeSerialData:incoming:
VFIBarcodeDelegate-p, 188
barcodeSerialNumber
VFIBarcode, 182
barcodeTriggerEvent:
VFIBarcodeDelegate-p, 188
barcodeTypeEnabled:
VFIBarcode, 162
barcodeTypes
VFIBarcode.h, 403
barcodeVersion
VFIBarcode, 182
batteryChargerStatus
VFIGen3DeviceInfo, 243
batteryCurrent
VFIGen3DeviceInfo, 243
batteryFullCharge
VFIGen3DeviceInfo, 244
batteryLevel
VFIControl, 211
batteryRemainingCharge
VFIGen3DeviceInfo, 244
batteryTemperature
VFIGen3DeviceInfo, 244
beepOnParsedScan:
VFIBarcode, 162

blackListCardSupportFlag
VFIEMVConfiguration, 229
BootloaderMajor
VFIKeypadVersion, 245
BootloaderMinor
VFIKeypadVersion, 245
btChannel
VFIAccessoryMgr, 156
btSerialData:incoming:
VFIPinpadDelegate-p, 364
bundleVersion
VFIGen3DeviceInfo, 244

C18:threshold:target:targetPercent:
VFIPinpad, 261
C19:value:
VFIPinpad, 261
C20:
VFIPinpad, 261
C21:
VFIPinpad, 261
C25:clear:numScripts:length:data:
VFIPinpad, 261
C30:language:amount:otherAmount:
VFIPinpad, 262
C30:language:amount:otherAmount:isRefundCTLS-
:disableCTLS:
VFIPinpad, 262
C32:otherAmount:merchantDecision:minRequestObjects:
VFIPinpad, 262
C32:otherAmount:merchantDecision:minRequestObjects-
:displayResult:
VFIPinpad, 262
C32:otherAmount:merchantDecision:minRequestObjects-
:displayResult:pinTryExceedStatus:display-
Amount:
VFIPinpad, 262
C34:displayResult:hostTags:
VFIPinpad, 263
C36:
VFIPinpad, 263
C52:otherAmount:merchantDecision:minRequestObjects:
VFIPinpad, 263
C54:displayResult:hostTags:
VFIPinpad, 263
COMM_RESULT_ACK
VFIBarcode.h, 403
COMM_RESULT_BAD_RESPONSE
VFIBarcode.h, 403
COMM_RESULT_BUSY
VFIBarcode.h, 403
COMM_RESULT_NAK
VFIBarcode.h, 403
COMM_RESULT_NO_RESPONSE
VFIBarcode.h, 403
CMD_BAR_DEV_CLOSE
Barcode_host.h, 383
CMD_BAR_DEV_OPEN
Barcode_host.h, 383
CMD_BARCODE_DATA
Barcode_host.h, 384
CMD_BEEP_IMMEDIATE
Barcode_host.h, 384
CMD_DEF, 145
command, 145
length, 145
protoVer, 145
CMD_EN_XMIT_FMT
Barcode_host.h, 384
CMD_MULTI_SCAN
Barcode_host.h, 384
CMD_PASS_THRU
Barcode_host.h, 384
CMD_PREFIX_VAL
Barcode_host.h, 384
CMD_SCAN_TIMEOUT
Barcode_host.h, 384
CMD_SET_TRIG_MODE
Barcode_host.h, 384
CMD_START_SCAN
Barcode_host.h, 384
CMD_STOP_SCAN
Barcode_host.h, 384
CMD_SUFFIX1_VAL
Barcode_host.h, 384
CMD_SUFFIX2_VAL
Barcode_host.h, 384
CMD_SYMBOLOGY
Barcode_host.h, 384
CODE_ID_AUS_POST
Barcode_host.h, 385
CODE_ID_CHINESE2OF5
Barcode_host.h, 385
CODE_ID_CODABAR
Barcode_host.h, 385
CODE_ID_CODE11
Barcode_host.h, 385
CODE_ID_CODE128_ISBT
Barcode_host.h, 385
CODE_ID_CODE39_32
Barcode_host.h, 385
CODE_ID_CODE93
Barcode_host.h, 385
CODE_ID_COUPONCODE
Barcode_host.h, 385
CODE_ID_CUECAT
Barcode_host.h, 385
CODE_ID_DISC2OF5

Barcode_host.h, 385
CODE_ID_GS1DATABAR
Barcode_host.h, 385
CODE_ID_GSI128
Barcode_host.h, 385
CODE_ID_INTL2OF5
Barcode_host.h, 385
CODE_ID_KOREAN3OF5
Barcode_host.h, 385
CODE_ID_MATRIX2OF5
Barcode_host.h, 385
CODE_ID_MAXICODE
Barcode_host.h, 385
CODE_ID_MSI
Barcode_host.h, 385
CODE_ID_NETH_KIX
Barcode_host.h, 385
CODE_ID_TRIOPTIC39
Barcode_host.h, 386
CODE_ID_UCCCOMPOS
Barcode_host.h, 386
CODE_ID_UK_POSTAL
Barcode_host.h, 386
CODE_ID_UPC_EA
Barcode_host.h, 386
CODE_ID_UPU_FICS
Barcode_host.h, 386
CODE_ID_US_PLANET
Barcode_host.h, 386
CODE_ID_USPS_4CB
Barcode_host.h, 386
CRData
AsyncSocket, 135
CRLFData
AsyncSocket, 135
canSafelySetDelegate
AsyncSocket, 134
cancelCommand
VFIPinpad, 263
cancelCommandWithDisplayOptions:
VFIPinpad, 264
cancelCommandWithDisplayOptions:andEMVFallback-:disableKeyboard:
VFIPinpad, 264
cancelPendingCommand
VFIPinpad, 265
cancelPendingCommandNoData
VFIPinpad, 265
capkIndex
VFIEMVConfiguration, 229
CardTypeAmex
swipeData.h, 396
VFISwipeData.h, 412
CardTypeDCIDisc
swipeData.h, 396, 397
VFISwipeData.h, 412, 413
CardTypeDiscover
swipeData.h, 396, 397
VFISwipeData.h, 412, 413
CardTypeEnRoute
swipeData.h, 396, 397
VFISwipeData.h, 412, 413
CardTypeJCBDisc
swipeData.h, 396, 397
VFISwipeData.h, 412, 413
CardTypeMC
swipeData.h, 396
VFISwipeData.h, 412
CardTypeNotRecognized
swipeData.h, 396
VFISwipeData.h, 412
CardTypeVisa
swipeData.h, 396
VFISwipeData.h, 412
cardAuthorization:otherAmount:merchantDecision:minRequestObjects:
VFIPinpad, 265
cardAuthorization:otherAmount:merchantDecision:minRequestObjects:displayResult:
VFIPinpad, 266
cardData
VFIFinancialData, 241
cardHolderName
VFICardData, 191
cardRemovalResult
VFIEMVCompletionData, 225
cardStatusDisplay
VFIEMVConfiguration, 230
CardType
swipeData.h, 396
VFISwipeData.h, 412
cardType
swipeData, 153
cardholderEntries
VFIFinancialData, 241
cashBackSelected
VFIDataEntry, 218
cashbackAmount
VFIDataEntry, 218
clear
decryptedCredentials, 146
ISCP_HIGH, 148
swipeData, 153
VFICardData, 191
VFICipheredData, 194
VFIDataEntry, 218
VFIDiagnostics, 220
VFIEMAFiles, 222

VFIEMVAuthorization, 224
VFIEMVCompletionData, 225
VFIEMVConfiguration, 229
VFIEMVResponse, 234
VFIEMVTags, 237
VFIEncryptedBarcode, 238
VFIEncryptionData, 239
VFIFinancialData, 241
VFIGen3DeviceInfo, 243
VFIKeypadVersion, 245
VFISoftwareVersion, 369
clearAllData
VFIPinpad, 272
clearBeforeScriptWrite
VFIEMVResponse, 235
clearCAPK
VFIPinpad, 272
clearDuplicatesBuffer
VFIBarcode, 163
clientID
decryptedCredentials, 146
closeAccessory:
VFIAccessoryMgr, 156
closeDevice
VFIBarcode, 163
VFIControl, 199
VFIPinpad, 273
closeDevices
VFIAccessoryMgr, 156
closeSession
VFIStream, 371
CloseSession_Gen20_Barcode
VFIAccessoryMgr.h, 399
CloseSession_Gen20_Control
VFIAccessoryMgr.h, 399
CloseSession_Gen20_XPI
VFIAccessoryMgr.h, 399
CloseSession_Gen20_Zontalk
VFIAccessoryMgr.h, 399
CloseSession_Gen25
VFIAccessoryMgr.h, 399
CloseSession_Gen30_Barcode
VFIAccessoryMgr.h, 399
CloseSession_Gen30_Control
VFIAccessoryMgr.h, 399
CloseSession_Gen30_XPI
VFIAccessoryMgr.h, 399
CloseSession_Gen30_Zontalk
VFIAccessoryMgr.h, 399
CloseSession_Printer
VFIAccessoryMgr.h, 399
CloseSession_Sled
VFIAccessoryMgr.h, 399
closeZontalk
VFIAccessoryMgr, 156
codeID
symb_INFO, 154
commResult
VFIBarcode.h, 403
command
CMD_DEF, 145
commandResult:
VFIBarcodeDelegate-p, 188
commandType
ISCP_HIGH, 149
completeOnlineEMV:displayResult:hostTags:
VFIPinpad, 277
configureBeep:
VFIBarcode, 163
connectToAddress:error:
AsyncSocket, 134
connectToAddress:viaInterfaceAddress:withTimeout-
:error:
AsyncSocket, 134
connectToAddress:withTimeout:error:
AsyncSocket, 134
connectToHost:onPort:error:
AsyncSocket, 135
connectToHost:onPort:withTimeout:error:
AsyncSocket, 135
connected
VFIBarcode, 182
VFIControl, 211
VFIPinpad, 359
Connected_Gen20_Barcode
VFIAccessoryMgr.h, 399
Connected_Gen20_Control
VFIAccessoryMgr.h, 399
Connected_Gen20_XPI
VFIAccessoryMgr.h, 399
Connected_Gen20_Zontalk
VFIAccessoryMgr.h, 399
Connected_Gen25
VFIAccessoryMgr.h, 399
Connected_Gen30_Barcode
VFIAccessoryMgr.h, 399
Connected_Gen30_Control
VFIAccessoryMgr.h, 399
Connected_Gen30_XPI
VFIAccessoryMgr.h, 399
Connected_Gen30_Zontalk
VFIAccessoryMgr.h, 399
Connected_Printer
VFIAccessoryMgr.h, 399
Connected_Sled
VFIAccessoryMgr.h, 399
connectedAddress
AsyncSocket, 134

connectedHost
AsyncSocket, 134
connectedPort
AsyncSocket, 134
connectedSockets
VFIPinpad, 359
consoleEnabled:
VFIBarcode, 163
VFIControl, 199
VFIPinpad, 279
continueTransactionTimeout
VFIPinpad, 359
controlBatteryEvent:batteryEvent:
VFIControlDelegate-p, 213
controlBatteryLevel:
VFIControlDelegate-p, 213
controlConnected
VFIControl, 211
controlConnected:
VFIControlDelegate-p, 214
controlDataReceived:
VFIControlDelegate-p, 214
controlDataSent:
VFIControlDelegate-p, 214
controlFirmwareRevision
VFIControl, 211
controlHardwareRevision
VFIControl, 211
controlInitialized:
VFIControlDelegate-p, 214
controlKeypadVersion:
VFIControlDelegate-p, 214
controlLogEntry:withSeverity:
VFIControlDelegate-p, 216
controlManufacturer
VFIControl, 211
controlModelNumber
VFIControl, 211
controlName
VFIControl, 211
controlNotfication:
VFIAccessoryMgr, 156
controlReconnectFinished
VFIControlDelegate-p, 216
controlReconnectStarted
VFIControlDelegate-p, 216
controlSerialData:incoming:
VFIControlDelegate-p, 216
controlSerialNumber
VFIControl, 211
controlSoftwareVersion:
VFIControlDelegate-p, 216
copyCAPKFile
VFIPinpad, 279
copyLastEMVTags
VFIPinpad, 279
copyPromptData
VFIPinpad, 280
copyStringResponse
VFIPinpad, 280
countryCode
VFIEMVConfiguration, 230
createCAPK:pkIndex:modulus:exponent:checksum-
:expiryDate:
VFIPinpad, 280
createCAPKFile:pkIndex:modulus:exponent:checksum-
:expiryDate:
VFIPinpad, 281
createMac:
VFIPinpad, 282
createTable:numRecords:
VFIPinpad, 282
ctlsOPTFLAG
VFIPinpad, 284
ctlsReaderFirmwareVersion
VFIDiagnostics, 220
ctlsState
VFIPinpad, 284
currencyCode
VFIEMVConfiguration, 230
currentSocket
VFIPinpad, 359

D01:
VFIPinpad, 285
D11:pkIndex:modulus:exponent:checksum:expiryDate:
VFIPinpad, 285
D11a:pkIndex:modulus:exponent:checksum:expiryDate:
VFIPinpad, 285
D12
VFIPinpad, 285
D14:tableData:
VFIPinpad, 286
D16:numRecords:
VFIPinpad, 286
D18:sig:CVM:
VFIPinpad, 286
D25:ctlsMode:pdolSupport:
VFIPinpad, 287
D25:ctlsMode:pdolSupport:valueLinkCtls:amexMode:
VFIPinpad, 287
D26:aid:version:
VFIPinpad, 288
D27
VFIPinpad, 288
D30:line2:line3:line4:amt1:amt2:amt3:amt4:
VFIPinpad, 288
DATA_SYNC_EN

POWER_STATUS, 151
DISABLE
Barcode_host.h, 386
dataEntryRequest:language:tip:cashback:surcharge:
VFIPinpad, 288
dataEntryRequestPrompt:
VFIPinpad, 289
DataReceivedNotification_Barcode
VFIAccessoryMgr.h, 399
DataReceivedNotification_Barcode_BT
VFIAccessoryMgr.h, 399
DataReceivedNotification_Control
VFIAccessoryMgr.h, 399
DataReceivedNotification_Control_BT
VFIAccessoryMgr.h, 399
DataReceivedNotification_Printer
VFIAccessoryMgr.h, 400
DataReceivedNotification_Sled
VFIAccessoryMgr.h, 400
DataReceivedNotification_XPI
VFIAccessoryMgr.h, 400
DataReceivedNotification_XPI_BT
VFIAccessoryMgr.h, 400
DataReceivedNotification_Zontalk
VFIAccessoryMgr.h, 400
DataReceivedNotification_Zontalk_BT
VFIAccessoryMgr.h, 400
dataType
VFICipheredData, 194
deactivateBarcodeType:
VFIBarcode, 163
decryptedCredentials, 145
clear, 146
clientID, 146
deviceKey, 146
deviceType, 147
serialNumber, 147
decryptedCredentials.h, 395
decryptedLength
VFIEncryptedBarcode, 238
defaultDDOL
VFIEMVConfiguration, 230
defaultTDOL
VFIEMVConfiguration, 230
delegate
AsyncSocket, 135
VFIBarcode, 182
VFIControl, 211
VFIPinpad, 359
VFIStream, 372
derivedKey:
VFIPinpad, 290
description
AsyncSocket, 135
deviceKey
decryptedCredentials, 146
deviceType
decryptedCredentials, 147
diagnosticInfo
VFIPinpad, 290
diagnosticInfoExtended
VFIPinpad, 290
disableBTBridge
VFIBTBridge, 189
disableBlocking
VFIBarcode, 164
VFIControl, 199
VFIPinpad, 291
disableKeypadOnUSBPower:
VFIControl, 199
disableMSR
VFIPinpad, 291
disableProtocol:
VFIBarcode, 164
VFIControl, 200
disabledBTBridge
VFIBTBridge, 189
disconnect
AsyncSocket, 135
disconnectAfterReading
AsyncSocket, 135
disconnectAfterReadingAndWriting
AsyncSocket, 135
disconnectAfterWriting
AsyncSocket, 135
Disconnected_Gen20_Barcode
VFIAccessoryMgr.h, 400
Disconnected_Gen20_Control
VFIAccessoryMgr.h, 400
Disconnected_Gen20_XPI
VFIAccessoryMgr.h, 400
Disconnected_Gen20_Zontalk
VFIAccessoryMgr.h, 400
Disconnected_Gen25
VFIAccessoryMgr.h, 400
Disconnected_Gen30_Barcode
VFIAccessoryMgr.h, 400
Disconnected_Gen30_Control
VFIAccessoryMgr.h, 400
Disconnected_Gen30_XPI
VFIAccessoryMgr.h, 400
Disconnected_Gen30_Zontalk
VFIAccessoryMgr.h, 400
Disconnected_Printer
VFIAccessoryMgr.h, 400
Disconnected_Sled
VFIAccessoryMgr.h, 400
displayMessage:clearScreen:

VFIPinpad, 291
displayMessage:withX:andY:shouldClearDisplay:
VFIControl, 200
displayMessageAndWait:line2:
VFIPinpad, 291
displayMessageAndWait:line2:line3:line4:
VFIPinpad, 292
displayMessagePromptInput:echo:timeout:maxLen:entryType:
VFIPinpad, 292
displayMessages:Line2:Line3:Line4:
VFIPinpad, 293
displayMessages:Line2:Line3:Line4:timeout:
VFIPinpad, 294
displayMessages:message1:message2:clearScreen:
VFIPinpad, 294
displayResult
VFIEMVResponse, 235
doInteracRequest:language:amount:pan:mac:
VFIPinpad, 294
dur
T_BEEP, 154
duration_1
M_BEEP_DEF, 150
duration_2
M_BEEP_DEF, 150
duration_3
M_BEEP_DEF, 150
duration_4
M_BEEP_DEF, 150
duration_5
M_BEEP_DEF, 150
duration_6
M_BEEP_DEF, 150

E00
VFIPinpad, 296
E02
VFIPinpad, 296
E04
VFIPinpad, 296
E06
VFIPinpad, 296
E08_RSA:publicKeyID:
VFIBarcode, 164
VFIPinpad, 296
E08_x509:
VFIBarcode, 164
VFIPinpad, 297
E10:
VFIBarcode, 164
VFIPinpad, 297
E10:keyIndex:
VFIPinpad, 297
E12:p75Data:
VFIBarcode, 164
VFIPinpad, 297
E14
VFIPinpad, 297
E20:
VFIPinpad, 297
ENABLE
Barcode_host.h, 386
ENABLE_414_419_434_439
Barcode_host.h, 386
ENABLE_977_SUPPLIMENT
Barcode_host.h, 386
EParms
VFICipheredData, 194
EXT_PWR
POWER_STATUS, 151
eaACC
VFIBarcode, 182
VFIControl, 211
VFIPinpad, 359
emaFiles
VFIPinpad, 359
emvAuthorization
VFIFinancialData, 241
emvTagData
VFIPinpad, 359
emvTags
VFICardData, 192
VFIEMVAuthorization, 224
VFIEMVCompletionData, 225
VFIEMVTags, 237
VFIFinancialData, 242
emvVersion
VFIDiagnostics, 220
enableBlocking
VFIBarcode, 164
VFIControl, 200
VFIPinpad, 298
enableLowBatteryNotification:
VFIControl, 200
enableLowBatteryNotification:withNotificationLevel:
VFIControl, 201
enableMSR
VFIPinpad, 298
enableMSRDualTrack
VFIPinpad, 298
enablePKI:
VFIBarcode, 165
enablePreBuffering
AsyncSocket, 136
enableSoftTrigger:
VFIBarcode, 165
enableWatchdogTimer:withCountdownValue:

VFIControl, 201
encryptedBlob_1
VFICipheredData, 194
encryptedBlob_2
VFICipheredData, 194
encryptedData
VFIEncryptedBarcode, 238
VFIFinancialData, 242
EncryptionMode_NOE
VFIPinpad.h, 411
EncryptionMode_PKI
VFIPinpad.h, 411
EncryptionMode_VSD
VFIPinpad.h, 411
EncryptionMode_VSP
VFIPinpad.h, 411
EncryptionMode
VFIPinpad.h, 410
encryptionType
VFICipheredData, 194
entryType
VFICardData, 192
excludeBarcodeType:
VFIBarcode, 165
executeCmd:
VFIBarcode, 165
expiryDate
VFICardData, 192

fallbackAllowedFlag
VFIEMVConfiguration, 230
fid
ISCP_HIGH, 149
FileName
VFIEMAFiles, 222
firmware
VFIBarcode, 183
FirmwareMajor
VFIKeypadVersion, 246
FirmwareMinor
VFIKeypadVersion, 246
firstName
swipeData, 153
floorLimit
VFIEMVConfiguration, 230
frameworkTimeout
VFIPinpad, 359
frameworkVersion
VFIBarcode, 166
VFIBTBridge, 189
VFIPinpad, 298
freq
T_BEEP, 154
freq_1
M_BEEP_DEF, 150
freq_2
M_BEEP_DEF, 150
freq_3
M_BEEP_DEF, 150
freq_4
M_BEEP_DEF, 150
freq_5
M_BEEP_DEF, 150
freq_6
M_BEEP_DEF, 150

GANG_CHARGER
POWER_STATUS, 151
GEN_PID_PASS_THRU
Barcode_host.h, 386
GID
VFIEMAFiles, 222
generalPrompt:message2:minLength:maxLength:language-:allowSkip:secureEntry:
VFIPinpad, 298
generalPrompt:message2:minLength:maxLength:language-:allowSkip:secureEntry:displayCanceled-:restoreScreen:
VFIPinpad, 299
generalPromptDefault:message2:minLength:maxLength-:language:initialValue:allowSkip:
VFIPinpad, 300
getBTFirmwareVersion
VFIControl, 202
getBTFriendlyName
VFIControl, 202
getBTPIN
VFIControl, 202
getBTSPPEnable
VFIControl, 202
getBarcode
VFIAccessoryMgr, 156
getBeep
VFIBarcode, 166
getCFReadStream
AsyncSocket, 136
getCFSocket
AsyncSocket, 136
getCFWriteStream
AsyncSocket, 136
getCardData:language:amount:otherAmount:
VFIPinpad, 300
getCardData:language:amount:otherAmount:isRefundCT-LS:disableCTLS:
VFIPinpad, 302
getCipheredData
VFIPinpad, 303
getControl

VFIAccessoryMgr, 156
getCrashInfo
VFIControl, 202
getEMVTag:
VFIPinpad, 305
getEMVTags:
VFIPinpad, 305
getEncryptionMode
VFIPinpad, 306
getFileHash:
VFIControl, 203
getGen3DeviceInfo
VFIControl, 203
getKSIVersion
VFIControl, 203
getKeyID
VFIPinpad, 306
getKeypadState
VFIControl, 203
getLogRecord
VFIControl, 204
getOSVersion
VFIPinpad, 306
getPKICipheredData
VFIPinpad, 307
getPowerStatus
VFIControl, 204
getPrinter
VFIAccessoryMgr, 156
getScanButtonMode
VFIBarcode, 166
getSled
VFIAccessoryMgr, 156
getVersion
VFIBarcode, 166
getXPI
VFIAccessoryMgr, 156
getXPIVersion
VFIPinpad, 307
getZontalk
VFIAccessoryMgr, 156
GlobalMarkup.d, 395
group
ISCP_HIGH, 149

HOST_BUF_SIZE
Barcode_host.h, 387
HOST_PWR
POWER_STATUS, 152
HOST_RX_BUF, 147
length, 147
sequence, 147
status, 147
HOST_TX_BUF, 147
length, 148
sequence, 148
hardwareVersion
VFIGen3DeviceInfo, 244
hostDecision
VFIEMVResponse, 235
hostPowerEnabled:
VFIControl, 204
hostPowerEnabled_e335:
VFIControl, 204
hostTagString
VFIEMVResponse, 235

I02
VFIPinpad, 307
IDEVICE_PRESENT
POWER_STATUS, 152
ISCP_HIGH, 148
clear, 148
commandType, 149
fid, 149
group, 149
param, 149
ISCP_HIGH.h, 396
ignoreDisconnect
VFIBarcode, 166
VFIControl, 205
VFIPinpad, 307
ignoreDuplicates:
VFIBarcode, 166
includeAllBarcodeTypes
VFIBarcode, 166
includeBarcodeType:
VFIBarcode, 167
init
AsyncSocket, 136
VFIBarcode, 167
VFIControl, 205
VFIPinpad, 307
initDevice
VFIBarcode, 167
VFIControl, 205
VFIPinpad, 308
initDeviceOnServer
VFIBarcode, 167
VFIControl, 205
initDeviceOnServer:port:
VFIPinpad, 308
initServerOnPort:
VFIPinpad, 309
initWithDelegate:
AsyncSocket, 136
initWithDelegate:userData:
AsyncSocket, 136

initializeContactless:sig:CVM:
VFIPinpad, 308
initializeIdle
VFIPinpad, 309
initialized
VFIBarcode, 183
VFIControl, 212
VFIPinpad, 360
ioTimer
VFIPinpad, 360
isBT
VFIStream, 371
isBTDevice
VFIAccessoryMgr, 156
isCTLS
VFICardData, 192
isConnected
AsyncSocket, 136
isGen3
VFIBarcode, 183
VFIControl, 212
VFIPinpad, 360
VFIStream, 371
isGen3Server
VFIPinpad, 309
isHostPowerEnabled
VFIControl, 205
isHostPowerEnabled_e335
VFIControl, 205
isIPv4
AsyncSocket, 136
isIPv6
AsyncSocket, 136
isRunning
VFIPinpad, 360
isVMFRemote
VFIPinpad, 360
issAcqFlag
VFIEMVConfiguration, 230
issuerReference
VFIEMVConfiguration, 230

keyID
VFICipheredData, 194
keyLoadErrorCode
VFIBarcode, 183
VFIPinpad, 360
keypadBeepEnabled:
VFIControl, 206
keypadEnabled:
VFIControl, 206

LFData
AsyncSocket, 136
lastName
swipeData, 153
length
CMD_DEF, 145
HOST_RX_BUF, 147
HOST_TX_BUF, 148
PARM_DEF, 151
listenSocket
VFIPinpad, 360
loadKeyInformation_RSA:publicKeyID:
VFIBarcode, 167
VFIPinpad, 309
loadKeyInformation_x509:
VFIBarcode, 168
VFIPinpad, 310
loadPKISig:p75Data:
VFIBarcode, 168
VFIPinpad, 310
localAddress
AsyncSocket, 136
localHost
AsyncSocket, 136
localPort
AsyncSocket, 136
logEnabled:
VFIBarcode, 169
VFIControl, 206
VFIPinpad, 311
Logical
VFIEMAFiles, 222

M_BEEP_DEF, 149
duration_1, 150
duration_2, 150
duration_3, 150
duration_4, 150
duration_5, 150
duration_6, 150
freq_1, 150
freq_2, 150
freq_3, 150
freq_4, 150
freq_5, 150
freq_6, 150
pause_1, 150
pause_2, 150
pause_3, 150
pause_4, 150
pause_5, 150
pause_6, 150
mAimPattern:
VFIBarcode, 169
mAppVersion
VFIBarcode, 169
mAutoBeepConfig:

VFIBarcode, 169
mAutoBeepConfig:scanBeep:errorBeep:
VFIBarcode, 170
mBeepImmediate:
VFIBarcode, 170
mButtonStatus
VFIBarcode, 170
mContinuousMode:
VFIBarcode, 171
mDeviceClose
VFIBarcode, 171
mDeviceOpen
VFIBarcode, 171
mDeviceOpenForce
VFIBarcode, 171
mDifferentSymbolDecodesTimeout:
VFIBarcode, 171
mDisableAllSymbologies
VFIBarcode, 171
mElectronicDisplayRead:
VFIBarcode, 171
mFirmwareVersion
VFIBarcode, 172
mGetAutoBeepError
VFIBarcode, 172
mGetAutoBeepMode
VFIBarcode, 172
mGetAutoBeepScan
VFIBarcode, 172
mPicklistMode:
VFIBarcode, 173
mRestoreDefaults
VFIBarcode, 173
mScanDataPrefix:value:
VFIBarcode, 173
mScanDataSuffix1:value:
VFIBarcode, 173
mScanDataSuffix2:value:
VFIBarcode, 173
mScanDataTransmitFormat:
VFIBarcode, 174
mScanTimeout:
VFIBarcode, 174
mStartScan
VFIBarcode, 174
mStopScan
VFIBarcode, 174
mSymbology:data:
VFIBarcode, 174
mSymbology:value:
VFIBarcode, 175
mTriggerMode:
VFIBarcode, 175
mUniqueCodeReport:
VFIBarcode, 175
macBlock
VFIEMVResponse, 235
VFIEncryptionData, 240
macData
VFIEMVResponse, 235
maskedPAN
swipeData, 153
maxTargetRSPercent
VFIEMVConfiguration, 230
menuChoiceSelection:message2:choice1:choice2:choice3-
:choice4:
VFIPinpad, 311
merchantCategoryCode
VFIEMVConfiguration, 230
merchantForceOnlineFlag
VFIEMVConfiguration, 231
merchantID
VFIEMVConfiguration, 231
modelNumber
VFIGen3DeviceInfo, 244
modifyCandidateListFlag
VFIEMVConfiguration, 231
moveToRunLoop:
AsyncSocket, 136

NAME
VFIEMAFiles, 223
noDisconnect
VFIBTBridge, 189
noDisplaySupportFlag
VFIEMVConfiguration, 231
noResponseNextCommand
VFIPinpad, 312
notConnected
VFIBTBridge, 189
numberOfScripts
VFIEMVResponse, 235

OSBuildDay
VFISoftwareVersion, 369
OSBuildMonth
VFISoftwareVersion, 369
OSBuildYear
VFISoftwareVersion, 370
OSID
VFISoftwareVersion, 370
OSPlatform
VFISoftwareVersion, 370
OSSubVersion
VFISoftwareVersion, 370
OSVersion
VFISoftwareVersion, 370
obtainCardData:
VFIPinpad, 312

obtainCardData:displayIdle:
VFIPinpad, 313
obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:
VFIPinpad, 314
obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout:
VFIPinpad, 315
obtainTrack2Data:optionalAmount:
VFIPinpad, 316
onSocket:didAcceptNewSocket:
AsyncSocketDelegate-p, 143
onSocket:didConnectToHost:port:
AsyncSocketDelegate-p, 143
onSocket:didReadData:withTag:
AsyncSocketDelegate-p, 143
onSocket:didReadPartialDataOfLength:tag:
AsyncSocketDelegate-p, 143
onSocket:didWriteDataWithTag:
AsyncSocketDelegate-p, 143
onSocket:didWritePartialDataOfLength:tag:
AsyncSocketDelegate-p, 143
onSocket:shouldTimeoutReadWithTag:elapsed:bytesDone:
AsyncSocketDelegate-p, 143
onSocket:shouldTimeoutWriteWithTag:elapsed:bytesDone:
AsyncSocketDelegate-p, 144
onSocket:wantsRunLoopForNewSocket:
AsyncSocketDelegate-p, 144
onSocket:willDisconnectWithError:
AsyncSocketDelegate-p, 144
onSocketDidDisconnect:
AsyncSocketDelegate-p, 144
onSocketDidSecure:
AsyncSocketDelegate-p, 144
onSocketWillConnect:
AsyncSocketDelegate-p, 145
openSession
VFIStream, 371
OpenSession_Gen20_Barcode
VFIAccessoryMgr.h, 400
OpenSession_Gen20_Control
VFIAccessoryMgr.h, 400
OpenSession_Gen20_XPI
VFIAccessoryMgr.h, 400
OpenSession_Gen20_Zontalk
VFIAccessoryMgr.h, 400
OpenSession_Gen25
VFIAccessoryMgr.h, 400
OpenSession_Gen25_Barcode
VFIAccessoryMgr.h, 400
OpenSession_Gen25_Control
VFIAccessoryMgr.h, 400
OpenSession_Gen25_XPI
VFIAccessoryMgr.h, 400
OpenSession_Gen25_Zontalk
VFIAccessoryMgr.h, 400
OpenSession_Gen30_Barcode
VFIAccessoryMgr.h, 401
OpenSession_Gen30_Control
VFIAccessoryMgr.h, 401
OpenSession_Gen30_XPI
VFIAccessoryMgr.h, 401
OpenSession_Gen30_Zontalk
VFIAccessoryMgr.h, 401
OpenSession_Printer
VFIAccessoryMgr.h, 401
OpenSession_Sled
VFIAccessoryMgr.h, 401
osversion
VFIDiagnostics, 220

PARM_DEF, 150
length, 151
pId, 151
PINTimeout
VFIPinpad, 361
pId
PARM_DEF, 151
POWER_STATUS, 151
EXT_PWR, 151
HOST_PWR, 152
POWERSHARE_ON
POWER_STATUS, 152
PS01:
VFIPinpad, 321
PS02:
VFIPinpad, 321
param
ISCP_HIGH, 149
partNumber
VFIGen3DeviceInfo, 244
partialReadBuffer
AsyncSocket, 141
pause_1
M_BEEP_DEF, 150
pause_2
M_BEEP_DEF, 150
pause_3
M_BEEP_DEF, 150
pause_4
M_BEEP_DEF, 150
pause_5
M_BEEP_DEF, 150
pause_6
M_BEEP_DEF, 150
performDisplayDiagnostic

VFIControl, 206
performKeypadAllKeyDiagnostic
VFIControl, 207
performMSRDiagnostic
VFIControl, 207
performPINEntryDiagnostic
VFIControl, 207
performTGKUpdateRequest
VFIPinpad, 317
pinBlock
VFIDataEntry, 218
VFIEncryptionData, 240
pinBlockStr
VFIEncryptionData, 240
pinBypass:
VFIPinpad, 317
pinBypassFlag
VFIEMVConfiguration, 231
pinDevriKey
VFIEMVConfiguration, 231
pinDevriMacro
VFIEMVConfiguration, 231
pinFormat
VFIEMVConfiguration, 231
pinMacroNumber
VFIEMVConfiguration, 231
pinManagementCardAuthorization:otherAmount:merchant-Decision:minRequestObjects:
VFIPinpad, 318
pinManagementCompleteOnlineEMV:displayResult:host-Tags:
VFIPinpad, 319
pinPadInitialized:
VFIPinpadDelegate-p, 365
pinScriptNumber
VFIEMVConfiguration, 231
pinSuccessful
VFIEMVAuthorization, 224
pinTimeout
VFIEMVConfiguration, 231
pinpadConnected
VFIPinpad, 360
pinpadConnected:
VFIPinpadDelegate-p, 364
pinpadDataReceived:
VFIPinpadDelegate-p, 364
pinpadDataSent:
VFIPinpadDelegate-p, 364
pinpadDownloadBlocks:sent:
VFIPinpadDelegate-p, 364
pinpadDownloadInfo:
VFIPinpadDelegate-p, 365
pinpadFirmwareRevision
VFIPinpad, 361
pinpadHardwareRevision
VFIPinpad, 361
pinpadLogEntry:withSeverity:
VFIPinpadDelegate-p, 365
pinpadMSRData:expMonth:expYear:track1Data:track2-Data:
VFIPinpadDelegate-p, 365
pinpadMSRData:expMonth:expYear:trackData:
VFIPinpadDelegate-p, 365
pinpadManufacturer
VFIPinpad, 361
pinpadMessage:
VFIPinpadDelegate-p, 365
pinpadModelNumber
VFIPinpad, 361
pinpadName
VFIPinpad, 361
pinpadNotfication:
VFIAccessoryMgr, 156
pinpadReconnectFinished
VFIPinpadDelegate-p, 367
pinpadReconnectStarted
VFIPinpadDelegate-p, 367
pinpadResultCode:result:
VFIPinpadDelegate-p, 367
pinpadSerialData:incoming:
VFIPinpadDelegate-p, 367
pinpadSerialNumber
VFIDiagnostics, 220
VFIPinpad, 361
pinpadServerData:incoming:
VFIPinpadDelegate-p, 367
postClosing
VFIAccessoryMgr, 156
powerDown
VFIControl, 207
processReceivedData:
VFIBarcode, 176
VFIControl, 207
VFIPinpad, 321
Processor
VFIKeypadVersion, 246
progressOfReadReturningTag:bytesDone:total:
AsyncSocket, 137
progressOfWriteReturningTag:bytesDone:total:
AsyncSocket, 137
protoVer
CMD_DEF, 145
Protocol_Gen20_Barcode
VFIAccessoryMgr.h, 401
Protocol_Gen20_Control
VFIAccessoryMgr.h, 401
Protocol_Gen20_XPI
VFIAccessoryMgr.h, 401

Protocol_Gen20_Zontalk
VFIAccessoryMgr.h, 401
Protocol_Gen25
VFIAccessoryMgr.h, 401
Protocol_Gen30_Barcode
VFIAccessoryMgr.h, 401
Protocol_Gen30_Control
VFIAccessoryMgr.h, 401
Protocol_Gen30_XPI
VFIAccessoryMgr.h, 401
Protocol_Gen30_Zontalk
VFIAccessoryMgr.h, 401
Protocol_Printer
VFIAccessoryMgr.h, 401
Protocol_Sled
VFIAccessoryMgr.h, 401
protocolString
VFIStream, 372
publicKey
VFIEncryptedBarcode, 238

Q40
VFIPinpad, 321
Q41
VFIPinpad, 321
Q42
VFIPinpad, 322
queryBatteryLevel
VFIControl, 207
queryDeviceLogging
VFIControl, 207
queryDiagnosticMode
VFIControl, 208
queryKeypadVersion
VFIControl, 208
querySoftwareVersion
VFIControl, 208

RESP_ACK
Barcode_host.h, 387
RESP_BARCODE_DATA
Barcode_host.h, 387
RESP_BUTTON_STATUS
Barcode_host.h, 387
RESP_NAK
Barcode_host.h, 387
readBytesAvailable
VFIStream, 371
readData:
VFIStream, 371
readDataToData:withTimeout:buffer:bufferOffset:max-
Length:tag:
AsyncSocket, 137
readDataToData:withTimeout:buffer:bufferOffset:tag:
AsyncSocket, 137
readDataToData:withTimeout:maxLength:tag:
AsyncSocket, 138
readDataToData:withTimeout:tag:
AsyncSocket, 138
readDataToLength:withTimeout:buffer:bufferOffset:tag:
AsyncSocket, 138
readDataToLength:withTimeout:tag:
AsyncSocket, 139
readDataWithTimeout:buffer:bufferOffset:maxLength:tag:
AsyncSocket, 139
readDataWithTimeout:buffer:bufferOffset:tag:
AsyncSocket, 139
readDataWithTimeout:tag:
AsyncSocket, 139
rebootDevice
VFIControl, 208
receivedData:protocol:
VFIStreamDelegate-p, 373
remoteMode
VFIPinpad, 361
removeCard
VFIPinpad, 322
removeRunLoopMode:
AsyncSocket, 139
requestEMAFiles
VFIPinpad, 322
requestPINRentry:
VFIPinpad, 323
requestPINRentry:pinpadPrompt:timeout:
VFIPinpad, 323
requestStringInput:timeout:maxLen:
VFIPinpad, 324
requestStringInput:timeout:maxLen:entryType:
VFIPinpad, 324
resetScannerToFactoryDefaults
VFIBarcode, 176
restartLoopDelay:
VFIBarcode, 176
VFIControl, 208
VFIPinpad, 325
retrieveEParmsData
VFIPinpad, 325
rfu1String
VFIEMVConfiguration, 232
rfu2String
VFIEMVConfiguration, 232
rfu3String
VFIEMVConfiguration, 232
rsThreshold
VFIEMVConfiguration, 232
runLoopModes
AsyncSocket, 139

S00

VFIPinpad, 327
S01
VFIPinpad, 327
S07:pin:balance:
VFIPinpad, 327
S11:line2:
VFIPinpad, 327
S11:line2:line3:line4:
VFIPinpad, 327
S12:message2:minLength:maxLength:language:allowSkip:secureEntry:
VFIPinpad, 327
S12:message2:minLength:maxLength:language:allowSkip:secureEntry:displayCanceled:restoreScreen:
VFIPinpad, 328
S12Default:message2:minLength:maxLength:language:initialValue:allowSkip:
VFIPinpad, 328
S13:message2:choice1:choice2:choice3:choice4:
VFIPinpad, 328
S14:Line2:Line3:Line4:
VFIPinpad, 328
S14:Line2:Line3:Line4:timeout:
VFIPinpad, 328
S16:
VFIPinpad, 329
S16:displayIdle:
VFIPinpad, 329
S16:displayIdle:displayExpiry:luhnCheck:entryMode:
VFIPinpad, 329
S16:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout:
VFIPinpad, 329
s16track1
VFICardData, 192
s16track2
VFICardData, 192
S20:optionalAmount:
VFIPinpad, 329
S21:language:tip:cashback:surcharge:
VFIPinpad, 329
S21prompt:
VFIPinpad, 330
S66:
VFIPinpad, 330
S67:
VFIPinpad, 330
S67:pinpadPrompt:timeout:
VFIPinpad, 331
S68
VFIPinpad, 331
S70:language:amount:pan:mac:
VFIPinpad, 332
S71:language:pinKey:macKey:msgLine1:msgLine2:macData:
VFIPinpad, 332
S71Fields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK:
VFIPinpad, 332
S95
VFIPinpad, 332
S95Extended
VFIPinpad, 332
S97
VFIPinpad, 332
SEV_DEBUG
VFIPinpad.h, 411
SEV_ERROR
VFIPinpad.h, 411
SEV_FATAL
VFIPinpad.h, 411
SEV_INFO_PINPAD
VFIPinpad.h, 411
SEV_UNKOWN
VFIPinpad.h, 411
SEV_WARN
VFIPinpad.h, 411
SSLCLIVersion
VFIGen3DeviceInfo, 244
SYM_ID_BOOKLAND
Barcode_host.h, 387
SYM_ID_CCA_EAN128
Barcode_host.h, 387
SYM_ID_CCA_EAN13
Barcode_host.h, 388
SYM_ID_CCA_EAN8
Barcode_host.h, 388
SYM_ID_CCA_RSS14
Barcode_host.h, 388
SYM_ID_CCA_UPCA
Barcode_host.h, 388
SYM_ID_CCA_UPCE
Barcode_host.h, 388
SYM_ID_CCB_EAN128
Barcode_host.h, 388
SYM_ID_CCB_EAN13
Barcode_host.h, 388
SYM_ID_CCB_EAN8
Barcode_host.h, 388
SYM_ID_CCB_RSS14
Barcode_host.h, 388
SYM_ID_CCB_UPCA
Barcode_host.h, 388
SYM_ID_CCB_UPCE
Barcode_host.h, 388
SYM_ID_CCC_EAN128
Barcode_host.h, 388

SYM_ID_CODABAR
Barcode_host.h, 388
SYM_ID_CODE11
Barcode_host.h, 388
SYM_ID_CODE128
Barcode_host.h, 388
SYM_ID_CODE32
Barcode_host.h, 388
SYM_ID_CODE39
Barcode_host.h, 388
SYM_ID_CODE39FULL
Barcode_host.h, 388
SYM_ID_CODE93
Barcode_host.h, 388
SYM_ID_COUPONCODE
Barcode_host.h, 388
SYM_ID_CUECAT
Barcode_host.h, 388
SYM_ID_D25
Barcode_host.h, 388
SYM_ID_DATAMATRIX
Barcode_host.h, 389
SYM_ID_EAN128
Barcode_host.h, 389
SYM_ID_EAN13
Barcode_host.h, 389
SYM_ID_EAN13_2
Barcode_host.h, 389
SYM_ID_EAN13_5
Barcode_host.h, 389
SYM_ID_EAN8
Barcode_host.h, 389
SYM_ID_EAN8_2
Barcode_host.h, 389
SYM_ID_EAN8_5
Barcode_host.h, 389
SYM_ID_IATA
Barcode_host.h, 389
SYM_ID_ISBT128
Barcode_host.h, 389
SYM_ID_ISBT128CONC
Barcode_host.h, 389
SYM_ID_ITF
Barcode_host.h, 389
SYM_ID_KOR3OF5
Barcode_host.h, 389
SYM_ID_MACROPDF
Barcode_host.h, 389
SYM_ID_MACROPDF417
Barcode_host.h, 389
SYM_ID_MAXICODE
Barcode_host.h, 389
SYM_ID_MICROPDF
Barcode_host.h, 389
SYM_ID_MSI
Barcode_host.h, 389
SYM_ID_PDF417
Barcode_host.h, 389
SYM_ID_PLANETUS
Barcode_host.h, 389
SYM_ID_POSTALAUST
Barcode_host.h, 389
SYM_ID_POSTALDUTCH
Barcode_host.h, 389
SYM_ID_POSTALJAPAN
Barcode_host.h, 389
SYM_ID_POSTALUK
Barcode_host.h, 389
SYM_ID_POSTBARCA
Barcode_host.h, 389
SYM_ID_POSTNETUS
Barcode_host.h, 389
SYM_ID_QRCODE
Barcode_host.h, 390
SYM_ID_RSS14
Barcode_host.h, 390
SYM_ID_RSSEXPAND
Barcode_host.h, 390
SYM_ID_RSSLIMIT
Barcode_host.h, 390
SYM_ID_SCANLETWEB
Barcode_host.h, 390
SYM_ID_TLC39
Barcode_host.h, 390
SYM_ID_TRIOPTIC
Barcode_host.h, 390
SYM_ID_UPCA
Barcode_host.h, 390
SYM_ID_UPCA_2
Barcode_host.h, 390
SYM_ID_UPCA_5
Barcode_host.h, 390
SYM_ID_UPCE
Barcode_host.h, 390
SYM_ID_UPCE1
Barcode_host.h, 390
SYM_ID_UPCE1_2
Barcode_host.h, 390
SYM_ID_UPCE1_5
Barcode_host.h, 390
SYM_ID_UPCE_2
Barcode_host.h, 390
SYM_ID_UPCE_5
Barcode_host.h, 390
SYM_PID_EN_AZTEC
Barcode_host.h, 391
SYM_PID_EN_CLSI
Barcode_host.h, 391

SYM_PID_EN_CODE11
Barcode_host.h, 391
SYM_PID_EN_CODE128
Barcode_host.h, 391
SYM_PID_EN_CODE39
Barcode_host.h, 391
SYM_PID_EN_CODE93
Barcode_host.h, 391
SYM_PID_EN_DISC2OF5
Barcode_host.h, 391
SYM_PID_EN_GS1128
Barcode_host.h, 391
SYM_PID_EN_ISBT
Barcode_host.h, 392
SYM_PID_EN_MSI
Barcode_host.h, 392
SYM_PID_EN_NOTIS
Barcode_host.h, 392
SYM_PID_EN_PDF417
Barcode_host.h, 392
SYM_PID_EN_UPCA
Barcode_host.h, 392
SYM_PID_EN_UPCE
Barcode_host.h, 392
SYM_PID_EN_UPCE1
Barcode_host.h, 392
SYM_PID_ISSN_EAN
Barcode_host.h, 393
scanIndex
VFIEncryptedBarcode, 238
scannerFirmwareVersion
VFIBarcode, 176
scanningModes
VFIBarcode.h, 403
schemeReference
VFIEMVConfiguration, 232
scriptData
VFIEMVResponse, 235
scriptId
VFIEMVResponse, 235
scriptLength
VFIEMVResponse, 235
selectEncryptionMode:
VFIPinpad, 333
selectEncryptionMode:keyIndex:
VFIPinpad, 333
sendBeep:
VFIBarcode, 176
sendCommand:
VFIBarcode, 176
VFIControl, 208
sendCommand:data:
VFIBarcode, 178
sendCommand:parameter:
VFIBarcode, 178
sendCommand:parameter:data:
VFIBarcode, 178
sendCommandLRC:
VFIControl, 209
sendData:
VFIBarcode, 178
VFIControl, 209
sendDataCommand:calcLRC:
VFIPinpad, 333
sendISCP:
VFIBarcode, 178
sendStringCommand:calcLRC:
VFIPinpad, 334
sendTriggerEvent:
VFIBarcode, 179
sequence
HOST_RX_BUF, 147
HOST_TX_BUF, 148
serialNumber
decryptedCredentials, 147
VFIDataEntry, 218
VFIEncryptionData, 240
VFIGen3DeviceInfo, 244
serverConnected:
VFIPinpadDelegate-p, 368
serverRunning
VFIPinpad, 361
serviceCode
VFICardData, 192
sessionOpen
VFIStream, 372
sessionOpen_Barcode
VFIStream, 372
sessionOpen_Control
VFIStream, 372
sessionOpen_XPI
VFIStream, 372
sessionOpen_Zontalk
VFIStream, 372
sessions
VFIAccessoryMgr, 156
setACKTimeout:
VFIPinpad, 334
setAccountEntryTimeout:
VFIPinpad, 334
setAdditionalEVMFlags:ctlsMode:pdolSupport:
VFIPinpad, 334
setAdditionalEVMFlags:ctlsMode:pdolSupport:valueLinkCtls:amexMode:
VFIPinpad, 335
setBTSessionOpen:
VFIStream, 371
setBeep:

VFIBarcode, 179
setBeepOff
VFIBarcode, 179
setBeepOn
VFIBarcode, 179
setC30CustomPrompts:line2:line3:line4:amt1:amt2:amt3-:amt4:
VFIPinpad, 336
setC30DefaultPrompts
VFIPinpad, 337
setContactlessLED:
VFIControl, 209
setContinueTransactionTimeout:
VFIPinpad, 337
setCurrencySymbol:
VFIPinpad, 338
setDateAndTime:
VFIControl, 209
setDelegate:
AsyncSocket, 140
setDeviceLogging:
VFIControl, 210
setEMVTag:value:
VFIPinpad, 338
setEMVTags:
VFIPinpad, 338
setEdge
VFIBarcode, 179
setFrameworkTimeout:
VFIPinpad, 339
setIOTimer:
VFIPinpad, 339
setInitLoop:
VFIBarcode, 179
VFIControl, 210
VFIPinpad, 339
setKSN20Char:
VFIPinpad, 339
setLevel
VFIBarcode, 180
setMultiScan
VFIBarcode, 180
setNoDisconnect:
VFIBTBridge, 189
setPINTimeout:
VFIPinpad, 341
setParameter:value:
VFIPinpad, 341
setParameterArray:
VFIPinpad, 341
setPassive
VFIBarcode, 180
setPromptTimeout:
VFIPinpad, 341
setROSParams:threshold:target:targetPercent:
VFIPinpad, 342
setRunLoopModes:
AsyncSocket, 140
setScanButtonMode:
VFIBarcode, 180
setScanTimeout:
VFIBarcode, 180
setScanner1D
VFIBarcode, 180
setScanner2D
VFIBarcode, 180
setSingleScan
VFIBarcode, 181
setSoft
VFIBarcode, 181
setTrigger:
VFIBarcode, 181
setUserData:
AsyncSocket, 140
setupControllerForAccessory:withProtocolString:
VFIStream, 371
sharedController
VFIAccessoryMgr, 156
VFIBarcode, 181
VFICardData, 191
VFICipheredData, 194
VFIControl, 210
VFIDataEntry, 218
VFIDiagnostics, 220
VFIEMAFiles, 222
VFIEMVAuthorization, 224
VFIEMVCompletionData, 225
VFIEMVTags, 237
VFIEncryptedBarcode, 238
VFIEncryptionData, 239
VFIGen3DeviceInfo, 243
VFIPinpad, 342
startDiagnosticMode
VFIControl, 210
startScan
VFIBarcode, 181
startServer
VFIPinpad, 342
startTLS:
AsyncSocket, 140
status
HOST_RX_BUF, 147
statusCTLS
VFIPinpad, 342
stopDiagnosticMode
VFIControl, 210
stopServer
VFIPinpad, 343

storeTimers:pin:balance:
VFIPinpad, 343
surchargeIndicator
VFIDataEntry, 218
svcInfoPartNumber
VFIDiagnostics, 220
svcInfoSerialNumber
VFIDiagnostics, 220
swipeData.h
CardTypeAmex, 396
CardTypeDCIDisc, 396, 397
CardTypeDiscover, 396, 397
CardTypeEnRoute, 396, 397
CardTypeJCBDisc, 396, 397
CardTypeMC, 396
CardTypeNotRecognized, 396
CardTypeVisa, 396
swipeData, 152
cardType, 153
clear, 153
firstName, 153
lastName, 153
maskedPAN, 153
swipeData.h, 396
CardType, 396
symb_AIM
Barcode_host.h, 395
symb_INFO, 153
AIMID, 154
codeID, 154
symbology, 154
symbology
symb_INFO, 154

TYPE_BARCODE_DATA
VFIBarcode.h, 405
TYPE_CONTROL
VFIBarcode.h, 404
TYPE_EVENT_NOTIFICATION
VFIBarcode.h, 405
TYPE_PERMISSION_READ
VFIBarcode.h, 404
TYPE_PERMISSION_WRITE
VFIBarcode.h, 404
TYPE_RESULT
VFIBarcode.h, 404
TYPE_SETUP_BARCODE_DATA
VFIBarcode.h, 405
TYPE_SETUP_PERMISSION_REPLY
VFIBarcode.h, 405
TYPE_SETUP_READ
VFIBarcode.h, 404
TYPE_SETUP_REPLY
VFIBarcode.h, 404
TYPE_SETUP_WRITE
VFIBarcode.h, 404
TYPE_STATUS_READ
VFIBarcode.h, 404
TYPE_STATUS_REPLY
VFIBarcode.h, 404
T_BEEP, 154
dur, 154
freq, 154
T_BEEP_DEF, 154
b1, 155
b2, 155
bPause, 155
TKG1_Track1
VFICipheredData, 194
TKG1_Track2
VFICipheredData, 195
TKG2_Track1
VFICipheredData, 195
TKG2_Track2
VFICipheredData, 195
TKG3_Track1
VFICipheredData, 195
TKG3_Track2
VFICipheredData, 195
TYPE
VFIBarcode.h, 404
tacDefault
VFIEMVConfiguration, 232
tacDenial
VFIEMVConfiguration, 232
tacOnline
VFIEMVConfiguration, 232
takKey
VFIEMVResponse, 235
targetRSPercent
VFIEMVConfiguration, 232
termCurExp
VFIEMVConfiguration, 232
terminalCapacity
VFIEMVConfiguration, 232
terminalCategoryCode
VFIEMVConfiguration, 233
terminalID
VFIEMVConfiguration, 233
terminalSerialNumber
VFIEncryptedBarcode, 238
terminalType
VFIEMVConfiguration, 233
testPINPad
VFIPinpad, 343
theConnectTimer
AsyncSocket, 141
theContext

AsyncSocket, 141
theCurrentRead
AsyncSocket, 141
theCurrentWrite
AsyncSocket, 141
theDelegate
AsyncSocket, 141
theFlags
AsyncSocket, 141
theNativeSocket4
AsyncSocket, 141
theNativeSocket6
AsyncSocket, 141
theReadQueue
AsyncSocket, 141
theReadStream
AsyncSocket, 141
theReadTimer
AsyncSocket, 141
theRunLoop
AsyncSocket, 141
theRunLoopModes
AsyncSocket, 142
theSocket4
AsyncSocket, 142
theSocket6
AsyncSocket, 142
theSource4
AsyncSocket, 142
theSource6
AsyncSocket, 142
theUserData
AsyncSocket, 142
theWriteQueue
AsyncSocket, 142
theWriteStream
AsyncSocket, 142
theWriteTimer
AsyncSocket, 142
tipAmount
VFIDataEntry, 218
tipSelected
VFIDataEntry, 218
tpkKey
VFIEMVResponse, 236
track1
VFICardData, 192
track2
VFICardData, 192
track3
VFICardData, 192
triggerResponse
VFIBarcode.h, 404
trmDataPresent
VFIEMVConfiguration, 233

unreadData
AsyncSocket, 141
unusedProtocols
VFIAccessoryMgr, 157
UpdateEA_Barcode
VFIAccessoryMgr.h, 401
UpdateEA_Control
VFIAccessoryMgr.h, 401
UpdateEA_Printer
VFIAccessoryMgr.h, 401
UpdateEA_Sled
VFIAccessoryMgr.h, 401
UpdateEA_XPI
VFIAccessoryMgr.h, 401
UpdateEA_Zontalk
VFIAccessoryMgr.h, 401
updateEMVConfigRecord:tableData:
VFIPinpad, 344
updateFromUrl:
VFIPinpad, 349
updateFromZip:
VFIPinpad, 350
updatePassThru:aid:version:
VFIPinpad, 351
userData
AsyncSocket, 141

VFIBarcode.h
BC_CODABAR, 403
BC_CODABLOCK, 403
BC_CODE11, 403
BC_CODE128, 403
BC_CODE39, 403
BC_CODE93, 403
BC_DATAMATRIX, 403
BC_EAN_UPC1, 403
BC_EAN_UPC2, 403
BC_GS1, 403
BC_INTERLEAVED_2_OF_5, 403
BC_MAXICODE, 403
BC_MSICODE, 403
BC_PDF417, 403
BC_PLESSYCODE, 403
BC_QRCODE, 403
BC_STAN_2_OF_5_2BARS, 403
BC_STAN_2_OF_5_3BARS, 403
BC_TELEPEN, 403
BCS_BUTTON_EDGE, 404
BCS_BUTTON_LEVEL, 404
BCS_BUTTON_PASSIVE, 404
BCS_BUTTON_SOFT, 404
BCS_TRIGGER_BOTH_PRESSED, 404
BCS_TRIGGER_LEFT_PRESSED, 404

BCS_TRIGGER_RELEASED, 404
BCS_TRIGGER_RIGHT_PRESSED, 404
COMM_RESULT_ACK, 403
COMM_RESULT_BAD_RESPONSE, 403
COMM_RESULT_BUSY, 403
COMM_RESULT_NAK, 403
COMM_RESULT_NO_RESPONSE, 403
TYPE_BARCODE_DATA, 405
TYPE_CONTROL, 404
TYPE_EVENT_NOTIFICATION, 405
TYPE_PERMISSION_READ, 404
TYPE_PERMISSION_WRITE, 404
TYPE_RESULT, 404
TYPE_SETUP_BARCODE_DATA, 405
TYPE_SETUP_PERMISSION_REPLY, 405
TYPE_SETUP_READ, 404
TYPE_SETUP_REPLY, 404
TYPE_SETUP_WRITE, 404
TYPE_STATUS_READ, 404
TYPE_STATUS_REPLY, 404
VFIPinpad.h
EncryptionMode_NOE, 411
EncryptionMode_PKI, 411
EncryptionMode_VSD, 411
EncryptionMode_VSP, 411
SEV_DEBUG, 411
SEV_ERROR, 411
SEV_FATAL, 411
SEV_INFO_PINPAD, 411
SEV_UNKOWN, 411
SEV_WARN, 411
VFIResultCode_ACK_Timeout, 411
VFIResultCode_Device_Not_Available, 411
VFIResultCode_Framework_Invalid_Data, 411
VFIResultCode_Framework_No_Blocking, 411
VFIResultCode_Framework_Timeout, 411
VFIResultCode_ACK_Timeout
VFIPinpad.h, 411
VFIResultCode_Device_Not_Available
VFIPinpad.h, 411
VFIResultCode_Framework_Invalid_Data
VFIPinpad.h, 411
VFIResultCode_Framework_No_Blocking
VFIPinpad.h, 411
VFIResultCode_Framework_Timeout
VFIPinpad.h, 411
VFISwipeData.h
CardTypeAmex, 412
CardTypeDCIDisc, 412, 413
CardTypeDiscover, 412, 413
CardTypeEnRoute, 412, 413
CardTypeJCBDisc, 412, 413
CardTypeMC, 412
CardTypeNotRecognized, 412
CardTypeVisa, 412
VERSION
VFIEMAFiles, 223
VFIAccessoryMgr, 155
_isBTDevice, 156
_sessions, 156
_unusedProtocols, 156
barcodeNotfication:, 156
btChannel, 156
closeAccessory:, 156
closeDevices, 156
closeZontalk, 156
controlNotfication:, 156
getBarcode, 156
getControl, 156
getPrinter, 156
getSled, 156
getXPI, 156
getZontalk, 156
isBTDevice, 156
pinpadNotfication:, 156
postClosing, 156
sessions, 156
sharedController, 156
unusedProtocols, 157
VFIAccessoryMgr.h, 397
CloseSession_Gen20_Barcode, 399
CloseSession_Gen20_Control, 399
CloseSession_Gen20_XPI, 399
CloseSession_Gen20_Zontalk, 399
CloseSession_Gen25, 399
CloseSession_Gen30_Barcode, 399
CloseSession_Gen30_Control, 399
CloseSession_Gen30_XPI, 399
CloseSession_Gen30_Zontalk, 399
CloseSession_Printer, 399
CloseSession_Sled, 399
Connected_Gen20_Barcode, 399
Connected_Gen20_Control, 399
Connected_Gen20_XPI, 399
Connected_Gen20_Zontalk, 399
Connected_Gen25, 399
Connected_Gen30_Barcode, 399
Connected_Gen30_Control, 399
Connected_Gen30_XPI, 399
Connected_Gen30_Zontalk, 399
Connected_Printer, 399
Connected_Sled, 399
DataReceivedNotification_Barcode, 399
DataReceivedNotification_Barcode_BT, 399
DataReceivedNotification_Control, 399
DataReceivedNotification_Control_BT, 399
DataReceivedNotification_Printer, 400
DataReceivedNotification_Sled, 400

DataReceivedNotification_XPI, 400
DataReceivedNotification_XPI_BT, 400
DataReceivedNotification_Zontalk, 400
DataReceivedNotification_Zontalk_BT, 400
Disconnected_Gen20_Barcode, 400
Disconnected_Gen20_Control, 400
Disconnected_Gen20_XPI, 400
Disconnected_Gen20_Zontalk, 400
Disconnected_Gen25, 400
Disconnected_Gen30_Barcode, 400
Disconnected_Gen30_Control, 400
Disconnected_Gen30_XPI, 400
Disconnected_Gen30_Zontalk, 400
Disconnected_Printer, 400
Disconnected_Sled, 400
OpenSession_Gen20_Barcode, 400
OpenSession_Gen20_Control, 400
OpenSession_Gen20_XPI, 400
OpenSession_Gen20_Zontalk, 400
OpenSession_Gen25, 400
OpenSession_Gen25_Barcode, 400
OpenSession_Gen25_Control, 400
OpenSession_Gen25_XPI, 400
OpenSession_Gen25_Zontalk, 400
OpenSession_Gen30_Barcode, 401
OpenSession_Gen30_Control, 401
OpenSession_Gen30_XPI, 401
OpenSession_Gen30_Zontalk, 401
OpenSession_Printer, 401
OpenSession_Sled, 401
Protocol_Gen20_Barcode, 401
Protocol_Gen20_Control, 401
Protocol_Gen20_XPI, 401
Protocol_Gen20_Zontalk, 401
Protocol_Gen25, 401
Protocol_Gen30_Barcode, 401
Protocol_Gen30_Control, 401
Protocol_Gen30_XPI, 401
Protocol_Gen30_Zontalk, 401
Protocol_Printer, 401
Protocol_Sled, 401
UpdateEA_Barcode, 401
UpdateEA_Control, 401
UpdateEA_Printer, 401
UpdateEA_Sled, 401
UpdateEA_XPI, 401
UpdateEA_Zontalk, 401
VFIBTBridge, 188
disableBTBridge, 189
disabledBTBridge, 189
frameworkVersion, 189
noDisconnect, 189
notConnected, 189
setNoDisconnect:, 189
VFIBTBridge.h, 405
VFIBarcode, 157
abortScan, 162
activateBarcodeType:, 162
BTconnected, 182
barcodeConnected, 181
barcodeFirmwareRevision, 182
barcodeHardwareRevision, 182
barcodeManufacturer, 182
barcodeModelNumber, 182
barcodeName, 182
barcodeSerialNumber, 182
barcodeTypeEnabled:, 162
barcodeVersion, 182
beepOnParsedScan:, 162
clearDuplicatesBuffer, 163
closeDevice, 163
configureBeep:, 163
connected, 182
consoleEnabled:, 163
deactivateBarcodeType:, 163
delegate, 182
disableBlocking, 164
disableProtocol:, 164
E08_RSA:publicKeyID:, 164
E08_x509:, 164
E10:, 164
E12:p75Data:, 164
eaACC, 182
enableBlocking, 164
enablePKI:, 165
enableSoftTrigger:, 165
excludeBarcodeType:, 165
executeCmd:, 165
firmware, 183
frameworkVersion, 166
getBeep, 166
getScanButtonMode, 166
getVersion, 166
ignoreDisconnect, 166
ignoreDuplicates:, 166
includeAllBarcodeTypes, 166
includeBarcodeType:, 167
init, 167
initDevice, 167
initDeviceOnServer, 167
initialized, 183
isGen3, 183
keyLoadErrorCode, 183
loadKeyInformation_RSA:publicKeyID:, 167
loadKeyInformation_x509:, 168
loadPKISig:p75Data:, 168
logEnabled:, 169
mAimPattern:, 169

mAppVersion, 169
mAutoBeepConfig:, 169
mAutoBeepConfig:scanBeep:errorBeep:, 170
mBeepImmediate:, 170
mButtonStatus, 170
mContinuousMode:, 171
mDeviceClose, 171
mDeviceOpen, 171
mDeviceOpenForce, 171
mDifferentSymbolDecodesTimeout:, 171
mDisableAllSymbologies, 171
mElectronicDisplayRead:, 171
mFirmwareVersion, 172
mGetAutoBeepError, 172
mGetAutoBeepMode, 172
mGetAutoBeepScan, 172
mPicklistMode:, 173
mRestoreDefaults, 173
mScanDataPrefix:value:, 173
mScanDataSuffix1:value:, 173
mScanDataSuffix2:value:, 173
mScanDataTransmitFormat:, 174
mScanTimeout:, 174
mStartScan, 174
mStopScan, 174
mSymbology:data:, 174
mSymbology:value:, 175
mTriggerMode:, 175
mUniqueCodeReport:, 175
processReceivedData:, 176
resetScannerToFactoryDefaults, 176
restartLoopDelay:, 176
scannerFirmwareVersion, 176
sendBeep:, 176
sendCommand:, 176
sendCommand:data:, 178
sendCommand:parameter:, 178
sendCommand:parameter:data:, 178
sendData:, 178
sendISCP:, 178
sendTriggerEvent:, 179
setBeep:, 179
setBeepOff, 179
setBeepOn, 179
setEdge, 179
setInitLoop:, 179
setLevel, 180
setMultiScan, 180
setPassive, 180
setScanButtonMode:, 180
setScanTimeout:, 180
setScanner1D, 180
setScanner2D, 180
setSingleScan, 181
setSoft, 181
setTrigger:, 181
sharedController, 181
startScan, 181
VFIBarcode.h, 402
barcodeTypes, 403
commResult, 403
scanningModes, 403
TYPE, 404
triggerResponse, 404
VFIBarcodeDelegate-p
barcodeConnected:, 185
barcodeDataReceived:, 186
barcodeDataSent:, 186
barcodeInitialized:, 186
barcodeLogEntry:withSeverity:, 186
barcodeMessage:, 186
barcodeReconnectFinished, 187
barcodeReconnectStarted, 187
barcodeScanData:, 187
barcodeScanData:barcodeType:, 187
barcodeScanData:code:symbol:aim:, 187
barcodeSerialData:incoming:, 188
barcodeTriggerEvent:, 188
commandResult:, 188
VFICardData, 190
AID, 191
accountNumber, 191
appLabel, 191
appPreferredName, 191
cardHolderName, 191
clear, 191
emvTags, 192
entryType, 192
expiryDate, 192
isCTLS, 192
s16track1, 192
s16track2, 192
serviceCode, 192
sharedController, 191
track1, 192
track2, 192
track3, 192
VFICardData.h, 405
VFICipheredData, 193
clear, 194
dataType, 194
EParms, 194
encryptedBlob_1, 194
encryptedBlob_2, 194
encryptionType, 194
keyID, 194
sharedController, 194
TKG1_Track1, 194

TKG1_Track2, 195
TKG2_Track1, 195
TKG2_Track2, 195
TKG3_Track1, 195
TKG3_Track2, 195
VSD_Blob, 195
VFICipheredData.h, 405
VFIControl, 195
BTconnected, 211
batteryLevel, 211
closeDevice, 199
connected, 211
consoleEnabled:, 199
controlConnected, 211
controlFirmwareRevision, 211
controlHardwareRevision, 211
controlManufacturer, 211
controlModelNumber, 211
controlName, 211
controlSerialNumber, 211
delegate, 211
disableBlocking, 199
disableKeypadOnUSBPower:, 199
disableProtocol:, 200
displayMessage:withX:andY:shouldClearDisplay:, 200
eaACC, 211
enableBlocking, 200
enableLowBatteryNotification:, 200
enableLowBatteryNotification:withNotificationLevel:, 201
enableWatchdogTimer:withCountdownValue:, 201
getBTFirmwareVersion, 202
getBTFriendlyName, 202
getBTPIN, 202
getBTSPPEnable, 202
getCrashInfo, 202
getFileHash:, 203
getGen3DeviceInfo, 203
getKSIVersion, 203
getKeypadState, 203
getLogRecord, 204
getPowerStatus, 204
hostPowerEnabled:, 204
hostPowerEnabled_e335:, 204
ignoreDisconnect, 205
init, 205
initDevice, 205
initDeviceOnServer, 205
initialized, 212
isGen3, 212
isHostPowerEnabled, 205
isHostPowerEnabled_e335, 205
keypadBeepEnabled:, 206
keypadEnabled:, 206
logEnabled:, 206
performDisplayDiagnostic, 206
performKeypadAllKeyDiagnostic, 207
performMSRDiagnostic, 207
performPINEntryDiagnostic, 207
powerDown, 207
processReceivedData:, 207
queryBatteryLevel, 207
queryDeviceLogging, 207
queryDiagnosticMode, 208
queryKeypadVersion, 208
querySoftwareVersion, 208
rebootDevice, 208
restartLoopDelay:, 208
sendCommand:, 208
sendCommandLRC:, 209
sendData:, 209
setContactlessLED:, 209
setDateAndTime:, 209
setDeviceLogging:, 210
setInitLoop:, 210
sharedController, 210
startDiagnosticMode, 210
stopDiagnosticMode, 210
vfiGen3DeviceInfo, 212
vfiKeypadVersion, 212
vfiSoftwareVersion, 212
VFIControl.h, 406
VFIControlDelegate-p
controlBatteryEvent:batteryEvent:, 213
controlBatteryLevel:, 213
controlConnected:, 214
controlDataReceived:, 214
controlDataSent:, 214
controlInitialized:, 214
controlKeypadVersion:, 214
controlLogEntry:withSeverity:, 216
controlReconnectFinished, 216
controlReconnectStarted, 216
controlSerialData:incoming:, 216
controlSoftwareVersion:, 216
VFIDataEntry, 216
accountSelected, 218
amountConfirmed, 218
cashBackSelected, 218
cashbackAmount, 218
clear, 218
pinBlock, 218
serialNumber, 218
sharedController, 218
surchargeIndicator, 218
tipAmount, 218
tipSelected, 218

VFIDataEntry.h, 406
VFIDecryptedCredentials.h, 406
VFIDiagnostics, 219
clear, 220
ctlsReaderFirmwareVersion, 220
emvVersion, 220
osversion, 220
pinpadSerialNumber, 220
sharedController, 220
svcInfoPartNumber, 220
svcInfoSerialNumber, 220
vspVersion, 220
vxciVersion, 221
xpiVersion, 221
VFIDiagnostics.h, 406
VFIEMAFiles, 221
ALLOW_MESSAGES, 222
clear, 222
FileName, 222
GID, 222
Logical, 222
NAME, 223
sharedController, 222
VERSION, 223
VFIEMAFiles.h, 407
VFIEMVAuthorization, 223
clear, 224
emvTags, 224
pinSuccessful, 224
sharedController, 224
VFIEMVAuthorization.h, 407
VFIEMVCompletionData, 224
cardRemovalResult, 225
clear, 225
emvTags, 225
sharedController, 225
validateMacResult, 226
writeToScriptResult, 226
VFIEMVCompletionData.h, 407
VFIEMVConfiguration, 227
acquireID, 229
additionalCapacity, 229
autoSelectApplication, 229
blackListCardSupportFlag, 229
capkIndex, 229
cardStatusDisplay, 230
clear, 229
countryCode, 230
currencyCode, 230
defaultDDOL, 230
defaultTDOL, 230
fallbackAllowedFlag, 230
floorLimit, 230
issAcqFlag, 230
issuerReference, 230
maxTargetRSPercent, 230
merchantCategoryCode, 230
merchantForceOnlineFlag, 231
merchantID, 231
modifyCandidateListFlag, 231
noDisplaySupportFlag, 231
pinBypassFlag, 231
pinDevriKey, 231
pinDevriMacro, 231
pinFormat, 231
pinMacroNumber, 231
pinScriptNumber, 231
pinTimeout, 231
rfu1String, 232
rfu2String, 232
rfu3String, 232
rsThreshold, 232
schemeReference, 232
tacDefault, 232
tacDenial, 232
tacOnline, 232
targetRSPercent, 232
termCurExp, 232
terminalCapacity, 232
terminalCategoryCode, 233
terminalID, 233
terminalType, 233
trmDataPresent, 233
vfi_rfu1, 233
vfi_rfu2, 233
vfi_rfu3, 233
VFIEMVConfiguration.h, 407
VFIEMVResponse, 233
clear, 234
clearBeforeScriptWrite, 235
displayResult, 235
hostDecision, 235
hostTagString, 235
macBlock, 235
macData, 235
numberOfScripts, 235
scriptData, 235
scriptId, 235
scriptLength, 235
takKey, 235
tpkKey, 236
VFIEMVResponse.h, 408
VFIEMVTags, 236
clear, 237
emvTags, 237
sharedController, 237
VFIEMVTags.h, 408
VFIEncryptedBarcode, 237

clear, 238
decryptedLength, 238
encryptedData, 238
publicKey, 238
scanIndex, 238
sharedController, 238
terminalSerialNumber, 238
VFIEncryptedBarcode.h, 408
VFIEncryptionData, 239
accountSelection, 240
clear, 239
macBlock, 240
pinBlock, 240
pinBlockStr, 240
serialNumber, 240
sharedController, 239
VFIEncryptionData.h, 408
VFIErrorCode
VFIPinpad.h, 411
VFIFinancialData, 240
additionalPromptResponses, 241
additionalPrompts, 241
cardData, 241
cardholderEntries, 241
clear, 241
emvAuthorization, 241
emvTags, 242
encryptedData, 242
VFIFinancialData.h, 409
VFIGen3DeviceInfo, 242
AMMVersion, 243
batteryChargerStatus, 243
batteryCurrent, 243
batteryFullCharge, 244
batteryRemainingCharge, 244
batteryTemperature, 244
bundleVersion, 244
clear, 243
hardwareVersion, 244
modelNumber, 244
partNumber, 244
SSLCLIVersion, 244
serialNumber, 244
sharedController, 243
XPIVersion, 244
VFIGen3DeviceInfo.h, 409
VFIKeypadVersion, 245
BootloaderMajor, 245
BootloaderMinor, 245
clear, 245
FirmwareMajor, 246
FirmwareMinor, 246
Processor, 246
VFIKeypadVersion.h, 409
VFIPinpad, 246
acceptEncryptPIN:, 258
accountEntryTimeout, 358
ackTimeout, 358
amountEntry:message1:message2:, 259
amountIntroduction:amountFormat:message1-:message2:, 260
amountResponse, 359
BTconnected, 359
C18:threshold:target:targetPercent:, 261
C19:value:, 261
C20:, 261
C21:, 261
C25:clear:numScripts:length:data:, 261
C30:language:amount:otherAmount:, 262
C30:language:amount:otherAmount:isRefundCTLS-:disableCTLS:, 262
C32:otherAmount:merchantDecision:minRequest-Objects:, 262
C32:otherAmount:merchantDecision:minRequest-Objects:displayResult:, 262
C34:displayResult:hostTags:, 263
C36:, 263
C52:otherAmount:merchantDecision:minRequest-Objects:, 263
C54:displayResult:hostTags:, 263
cancelCommand, 263
cancelCommandWithDisplayOptions:, 264
cancelCommandWithDisplayOptions:andEMV-Fallback:disableKeyboard:, 264
cancelPendingCommand, 265
cancelPendingCommandNoData, 265
cardAuthorization:otherAmount:merchantDecision-:minRequestObjects:, 265
cardAuthorization:otherAmount:merchantDecision-:minRequestObjects:displayResult:, 266
clearAllData, 272
clearCAPK, 272
closeDevice, 273
completeOnlineEMV:displayResult:hostTags:, 277
connected, 359
connectedSockets, 359
consoleEnabled:, 279
continueTransactionTimeout, 359
copyCAPKFile, 279
copyLastEMVTags, 279
copyPromptData, 280
copyStringResponse, 280
createCAPK:pkIndex:modulus:exponent:checksum-:expiryDate:, 280
createCAPKFile:pkIndex:modulus:exponent:checksum-:expiryDate:, 281
createMac:, 282
createTable:numRecords:, 282

ctlsOPTFLAG, 284
ctlsState, 284
currentSocket, 359
D01:, 285
D11:pkIndex:modulus:exponent:checksum:expiryDate:, 285
D11a:pkIndex:modulus:exponent:checksum:expiryDate:, 285
D12, 285
D14:tableData:, 286
D16:numRecords:, 286
D18:sig:CVM:, 286
D25:ctlsMode:pdolSupport:, 287
D25:ctlsMode:pdolSupport:valueLinkCtls:amexMode:, 287
D26:aid:version:, 288
D27, 288
D30:line2:line3:line4:amt1:amt2:amt3:amt4:, 288
dataEntryRequest:language:tip:cashback:surcharge:, 288
dataEntryRequestPrompt:, 289
delegate, 359
derivedKey:, 290
diagnosticInfo, 290
diagnosticInfoExtended, 290
disableBlocking, 291
disableMSR, 291
displayMessage:clearScreen:, 291
displayMessageAndWait:line2:, 291
displayMessageAndWait:line2:line3:line4:, 292
displayMessagePromptInput:echo:timeout:maxLen:entryType:, 292
displayMessages:Line2:Line3:Line4:, 293
displayMessages:Line2:Line3:Line4:timeout:, 294
displayMessages:message1:message2:clearScreen:, 294
doInteracRequest:language:amount:pan:mac:, 294
E00, 296
E02, 296
E04, 296
E06, 296
E08_RSA:publicKeyID:, 296
E08_x509:, 297
E10:, 297
E10:keyIndex:, 297
E12:p75Data:, 297
E14, 297
E20:, 297
eaACC, 359
emaFiles, 359
emvTagData, 359
enableBlocking, 298
enableMSR, 298
enableMSRDualTrack, 298
frameworkTimeout, 359
frameworkVersion, 298
generalPrompt:message2:minLength:maxLength:language:allowSkip:secureEntry:, 298
generalPromptDefault:message2:minLength:maxLength:language:initialValue:allowSkip:, 300
getCardData:language:amount:otherAmount:, 300
getCipheredData, 303
getEMVTag:, 305
getEMVTags:, 305
getEncryptionMode, 306
getKeyID, 306
getOSVersion, 306
getPKICipheredData, 307
getXPIVersion, 307
I02, 307
ignoreDisconnect, 307
init, 307
initDevice, 308
initDeviceOnServer:port:, 308
initServerOnPort:, 309
initializeContactless:sig:CVM:, 308
initializeIdle, 309
initialized, 360
ioTimer, 360
isGen3, 360
isGen3Server, 309
isRunning, 360
isVMFRemote, 360
keyLoadErrorCode, 360
listenSocket, 360
loadKeyInformation_RSA:publicKeyID:, 309
loadKeyInformation_x509:, 310
loadPKISig:p75Data:, 310
logEnabled:, 311
menuChoiceSelection:message2:choice1:choice2:choice3:choice4:, 311
noResponseNextCommand, 312
obtainCardData:, 312
obtainCardData:displayIdle:, 313
obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:, 314
obtainCardData:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout:, 315
obtainTrack2Data:optionalAmount:, 316
PINTimeout, 361
PS01:, 321
PS02:, 321
performTGKUpdateRequest, 317
pinBypass:, 317
pinManagementCardAuthorization:otherAmount:merchantDecision:minRequestObjects:, 318

pinManagementCompleteOnlineEMV:displayResult:hostTags:, 319
pinpadConnected, 360
pinpadFirmwareRevision, 361
pinpadHardwareRevision, 361
pinpadManufacturer, 361
pinpadModelNumber, 361
pinpadName, 361
pinpadSerialNumber, 361
processReceivedData:, 321
Q40, 321
Q41, 321
Q42, 322
remoteMode, 361
removeCard, 322
requestEMAFiles, 322
requestPINRentry:, 323
requestPINRentry:pinpadPrompt:timeout:, 323
requestStringInput:timeout:maxLen:, 324
requestStringInput:timeout:maxLen:entryType:, 324
restartLoopDelay:, 325
retrieveEParmsData, 325
S00, 327
S01, 327
S07:pin:balance:, 327
S11:line2:, 327
S11:line2:line3:line4:, 327
S12:message2:minLength:maxLength:language:allowSkip:secureEntry:, 327
S12:message2:minLength:maxLength:language:allowSkip:secureEntry:displayCanceled:restoreScreen:, 328
S12Default:message2:minLength:maxLength:language:initialValue:allowSkip:, 328
S13:message2:choice1:choice2:choice3:choice4:, 328
S14:Line2:Line3:Line4:, 328
S14:Line2:Line3:Line4:timeout:, 328
S16:, 329
S16:displayIdle:, 329
S16:displayIdle:displayExpiry:luhnCheck:entryMode:, 329
S16:displayIdle:displayExpiry:luhnCheck:entryMode:cvvPrompting:masking:timeout:, 329
S20:optionalAmount:, 329
S21:language:tip:cashback:surcharge:, 329
S21prompt:, 330
S66:, 330
S67:, 330
S67:pinpadPrompt:timeout:, 331
S68, 331
S70:language:amount:pan:mac:, 332
S71:language:pinKey:macKey:msgLine1:msgLine2:macData:, 332
S71Fields:language:msgLine1:msgLine2:macData:identity1:data1:identityK:dataK:, 332
S95, 332
S95Extended, 332
S97, 332
selectEncryptionMode:, 333
selectEncryptionMode:keyIndex:, 333
sendDataCommand:calcLRC:, 333
sendStringCommand:calcLRC:, 334
serverRunning, 361
setACKTimeout:, 334
setAccountEntryTimeout:, 334
setAdditionalEVMFlags:ctlsMode:pdolSupport:, 334
setAdditionalEVMFlags:ctlsMode:pdolSupport:valueLinkCtls:amexMode:, 335
setC30CustomPrompts:line2:line3:line4:amt1:amt2:amt3:amt4:, 336
setC30DefaultPrompts, 337
setContinueTransactionTimeout:, 337
setCurrencySymbol:, 338
setEMVTag:value:, 338
setEMVTags:, 338
setFrameworkTimeout:, 339
setIOTimer:, 339
setInitLoop:, 339
setKSN20Char:, 339
setPINTimeout:, 341
setParameter:value:, 341
setParameterArray:, 341
setPromptTimeout:, 341
setROSParams:threshold:target:targetPercent:, 342
sharedController, 342
startServer, 342
statusCTLS, 342
stopServer, 343
storeTimers:pin:balance:, 343
testPINPad, 343
updateEMVConfigRecord:tableData:, 344
updateFromUrl:, 349
updateFromZip:, 350
updatePassThru:aid:version:, 351
validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData:, 353
vfiCardData, 361
vfiCipheredData, 361
vfiDataEntry, 362
vfiDiagnostics, 362
vfiEMVAuthorization, 362
vfiEMVCompletionData, 362
vfiEMVTags, 362
vfiEncryptionData, 362
vsdKeyIndex, 362
waitForResponse:, 355
writeToScriptFile:clear:numScripts:length:data:, 355

writeToSock:, 357
Z2:clearScreen:, 357
Z3:message1:message2:clearScreen:, 357
Z50:timeout:maxLen:, 357
Z50:timeout:maxLen:entryType:, 357
Z52:echo:timeout:maxLen:entryType:, 358
Z56:message1:message2:, 358
Z58:amountFormat:message1:message2:, 358
Z60:, 358
VFIPinpad.h, 409
EncryptionMode, 410
VFIErrorCode, 411
VFIPinpadDelegate-p
btSerialData:incoming:, 364
pinPadInitialized:, 365
pinpadConnected:, 364
pinpadDataReceived:, 364
pinpadDataSent:, 364
pinpadDownloadBlocks:sent:, 364
pinpadDownloadInfo:, 365
pinpadLogEntry:withSeverity:, 365
pinpadMSRData:expMonth:expYear:track1Data-:track2Data:, 365
pinpadMSRData:expMonth:expYear:trackData:, 365
pinpadMessage:, 365
pinpadReconnectFinished, 367
pinpadReconnectStarted, 367
pinpadResultCode:result:, 367
pinpadSerialData:incoming:, 367
pinpadServerData:incoming:, 367
serverConnected:, 368
VFISoftwareVersion, 368
AppBuild, 369
AppMajor, 369
AppMinor, 369
clear, 369
OSBuildDay, 369
OSBuildMonth, 369
OSBuildYear, 370
OSID, 370
OSPlatform, 370
OSSubVersion, 370
OSVersion, 370
VFISoftwareVersion.h, 411
VFIStream, 370
_accessory, 372
_protocolString, 372
_readData, 372
_session, 372
_sessionOpen, 372
_sessionOpen_Barcode, 372
_sessionOpen_Control, 372
_sessionOpen_XPI, 372
_sessionOpen_Zontalk, 372
_writeData, 372
accessory, 372
closeSession, 371
delegate, 372
isBT, 371
isGen3, 371
openSession, 371
protocolString, 372
readBytesAvailable, 371
readData:, 371
sessionOpen, 372
sessionOpen_Barcode, 372
sessionOpen_Control, 372
sessionOpen_XPI, 372
sessionOpen_Zontalk, 372
setBTSessionOpen:, 371
setupControllerForAccessory:withProtocolString:, 371
writeData:channel:isOpen:, 371
VFIStream.h, 412
VFIStreamDelegate-p
receivedData:protocol:, 373
VFISwipeData.h, 412
CardType, 412
VSD_Blob
VFICipheredData, 195
validateMacResponse:language:pinKey:macKey:msgLine1:msgLine2:macData:
VFIPinpad, 353
validateMacResult
VFIEMVCompletionData, 226
vfi_rfu1
VFIEMVConfiguration, 233
vfi_rfu2
VFIEMVConfiguration, 233
vfi_rfu3
VFIEMVConfiguration, 233
vfiCardData
VFIPinpad, 361
vfiCipheredData
VFIPinpad, 361
vfiDataEntry
VFIPinpad, 362
vfiDiagnostics
VFIPinpad, 362
vfiEMVAuthorization
VFIPinpad, 362
vfiEMVCompletionData
VFIPinpad, 362
vfiEMVTags
VFIPinpad, 362
vfiEncryptionData
VFIPinpad, 362
vfiGen3DeviceInfo

VFIControl, 212
vfiKeypadVersion
VFIControl, 212
vfiSoftwareVersion
VFIControl, 212
vsdKeyIndex
VFIPinpad, 362
vspVersion
VFIDiagnostics, 220
vxciVersion
VFIDiagnostics, 221

waitForResponse:
VFIPinpad, 355
write:
AsyncSocket, 141
writeData:channel:isOpen:
VFIStream, 371
writeData:withTimeout:tag:
AsyncSocket, 141
writeToScriptFile:clear:numScripts:length:data:
VFIPinpad, 355
writeToScriptResult
VFIEMVCompletionData, 226
writeToSock:
VFIPinpad, 357

XPIVersion
VFIGen3DeviceInfo, 244
xpiVersion
VFIDiagnostics, 221

Z2:clearScreen:
VFIPinpad, 357
Z3:message1:message2:clearScreen:
VFIPinpad, 357
Z50:timeout:maxLen:
VFIPinpad, 357
Z50:timeout:maxLen:entryType:
VFIPinpad, 357
Z52:echo:timeout:maxLen:entryType:
VFIPinpad, 358
Z56:message1:message2:
VFIPinpad, 358
Z58:amountFormat:message1:message2:
VFIPinpad, 358
Z60:
VFIPinpad, 358
ZeroData
AsyncSocket, 141