

INPAS Crypto Module

Для разработчиков

Версия 1.0

История изменений документа

Версия	Дата	Перечень изменений
1.0	17.07.2020	Первоначальная версия

Содержание

1. Описание архитектуры платежного приложения «Unipos Terminal»	4
1.1. Описание уровней архитектуры платежного приложения «Unipos Terminal»	5
2. Уровень аппаратной абстракции и дополнительные библиотеки.....	7
2.1. Crypto Module.....	7
3. Приложения	11
3.1. Компонентная модель ПО «Unipos»	11
3.2. Компонентная модель системных библиотек «Unipos».....	12
3.3. Список заголовочных файлов	13

1. Описание архитектуры платежного приложения «Unipos Terminal»

Архитектура платежного приложения «Unipos Terminal» является многоуровневой и строится по принципу разделения кода на логические уровни, отвечающие за определенную категорию функционала. Каждый уровень имеет определенный и фиксированный интерфейс, что позволяет изменять конечную реализацию логического уровня без модификации других уровней.

Функции одного уровня, используемые несколькими загруженными приложениями, располагаются в отдельных библиотеках. Это позволяет уменьшить размер исполняемых приложений за счет исключения дублирования кода, а также оптимизировать использование памяти (Рисунок 1. Многоуровневая модель с отдельными библиотеками).

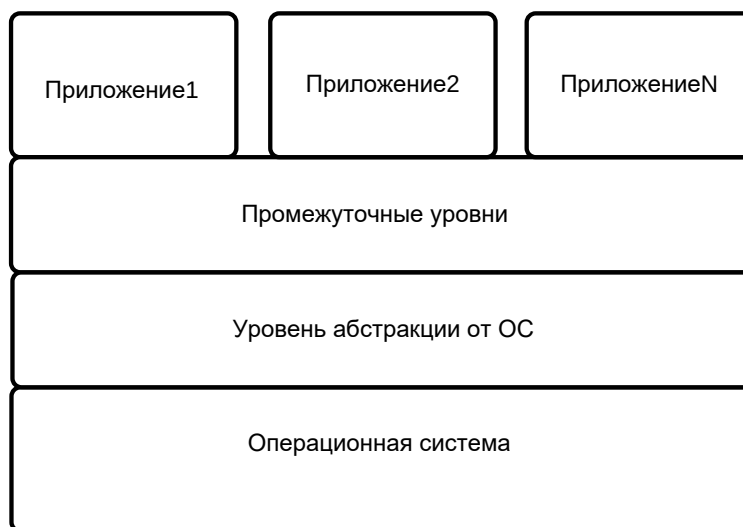


Рисунок 1. Многоуровневая модель с отдельными библиотеками

Если аппаратная платформа не поддерживает работу с разделяемыми библиотеками, то весь функционал приложения находится в одном исполняемом модуле (Рисунок 2. Многоуровневая модель без использования отдельных библиотек).

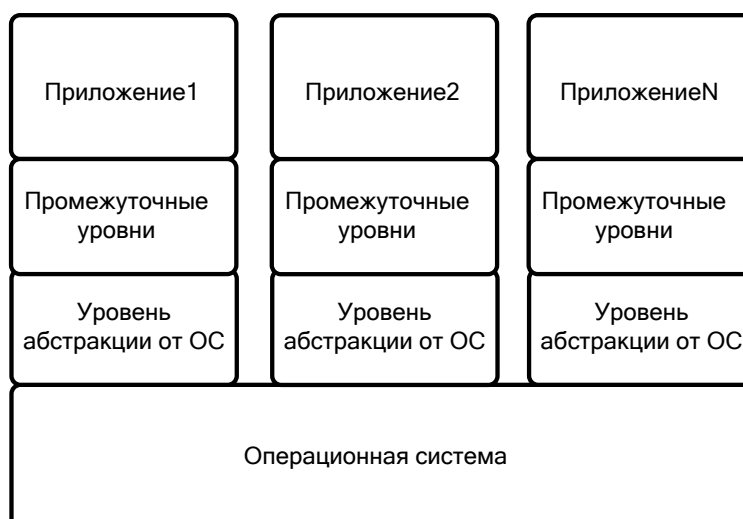


Рисунок 2. Многоуровневая модель без использования отдельных библиотек

1.1. Описание уровней архитектуры платежного приложения «Unipos Terminal»

Самый верхний уровень – это требуемая система (уровень «**Приложения**»), а самый нижний уровень – это доступные ресурсы: аппаратура, операционные системы, библиотеки (уровень «**Операционная система**»).

Обязательным уровнем является «**Уровень абстракции от операционной системы**», он обеспечивает переносимость системы на различные аппаратные платформы. Позволяет вышестоящим уровням абстрагироваться от конечного оборудования.

Интерфейс взаимодействия с устройствами через общий интерфейс включает в себя работу с:

- системой ввода/вывода;
- считывателями карт (магнитных, чиповых, бесконтактных);
- устройствами предоставления криптографического функционала;
- устройствами печати;
- коммуникационными устройствами;
- памятью;
- файловой системой.

Промежуточные уровни вводятся разработчиками системы. Каждый уровень базируется на всех следующих за ним уровнях. То есть уровень «**Приложения**» использует в своей работе уровни «**Промежуточные уровни**» и «**Уровень абстракции от операционной системы**». Использование уровня «**Операционная система**» допустимо только «**Уровнем абстракции от ОС**».

1. Уровень абстракции от железа («**Hardware Abstract Layer**»):

- a. **HAL** - это модуль абстракции, между физическим оборудованием POS-терминала и программным обеспечением, которое работает на этом терминале. Он реализует общий интерфейс для связи с оборудованием на устройствах «**VeriFone**» и «**PAX**»;
- b. **Communication** – модуль взаимодействия терминала с внешними устройствами по стандартным каналам связи. Модуль обеспечивает единый интерфейс работы с устройствами связи и транспортными протоколами для управляющего приложения;
- c. **VSS** – модуль реализующий взаимодействие со специализированными скриптами безопасности «**VeriShield Security Scripts**» и предназначен для поддержки различных схем управления ключами, выполнения автономных проверок PIN и других функций по работе с хранениями ключей безопасности в аппаратном пинпаде;
- d. **Crypto** - модуль, реализующий криптографические функции, такие как шифрование, дешифрование, расчет MAC, загрузка ключей в крипто-модуль (VSS/SAM), получение информации о загруженных ключах, преобразование форматов ключей (из «**ATM**» в «**GISKE**») и др.

2. Вспомогательные библиотеки («**Utils Layer**»):

- a. **Utils** – вспомогательный модуль, содержащий имплементацию функций по работе со строками, файловыми хранилищами, алгоритмическим преобразованиями и др.;
- b. **TACL** – модуль упаковки данных в конкретный формат хостовых сообщений;
- c. **OpenSSL** – модуль, реализующий функции по работе с криптографической библиотекой с открытым исходным кодом «**OpenSSL**», реализующая поддержку SSL/TLS;
- d. **IG** – графическая библиотека, предназначена для реализации графического интерфейса пользователя;

- e. **Memory Manager** – модуль распределения и управления доступной памятью (heap);
 - f. **DLA** – модуль, реализующий взаимодействие с системой удаленной загрузки мастер-ключей «RKL».
3. Сервисный уровень («Service Layer»):
- a. **Loyalty** – модуль, реализующий функциональность систем лояльности;
 - b. **ESLIP** – модуль взаимодействия с сервером хранения электронных чеков;
 - c. **HOSTS** – модуль взаимодействия с банковским хостом, реализующий протокол обмена данными;
 - d. **Monitoring** – модуль взаимодействия с сервером мониторинга, реализующий протокол обмена и регистрации событий на терминале;
 - e. **Plugin Manager** – интерфейс для взаимодействия терминального ПО с плагинами (динамические библиотеки реализующие вспомогательную функциональность).
4. Уровень приложения:
- a. **EMV** – платформо-независимый модуль, реализующий логику обслуживания EMV-карт;
 - b. **Contactless** – модуль обслуживания бесконтактных карт для взаимодействия с NFC-ридером по протоколу вендора;
 - c. **SmartSale** – модуль реализующий логику взаимодействия терминала и кассы в интегрированных решениях по протоколу «SA»;
 - d. **DLS** – модуль реализующий функциональность системы лояльности DLS;
 - e. **Host Client** – интерфейс между основным платежным приложением и хостовой библиотекой;
 - f. **TransactionExec** – реализация основной логики выполнения транзакций на терминале;
 - g. **Params** – модуль взаимодействия с системой параметризации.

2. Уровень аппаратной абстракции и дополнительные библиотеки

Программное обеспечение POS-терминала состоит из трех уровней:

1. Уровень программного обеспечения;
2. Уровень аппаратной абстракции и дополнительные библиотеки;
3. Уровень операционной системы.

Этот пункт описывает только второй уровень. Общее расположение и общее описание программного обеспечения POS-терминала показано на рисунке ниже (Рисунок 3. Общее расположение и общее описание программного обеспечения POS-терминала).

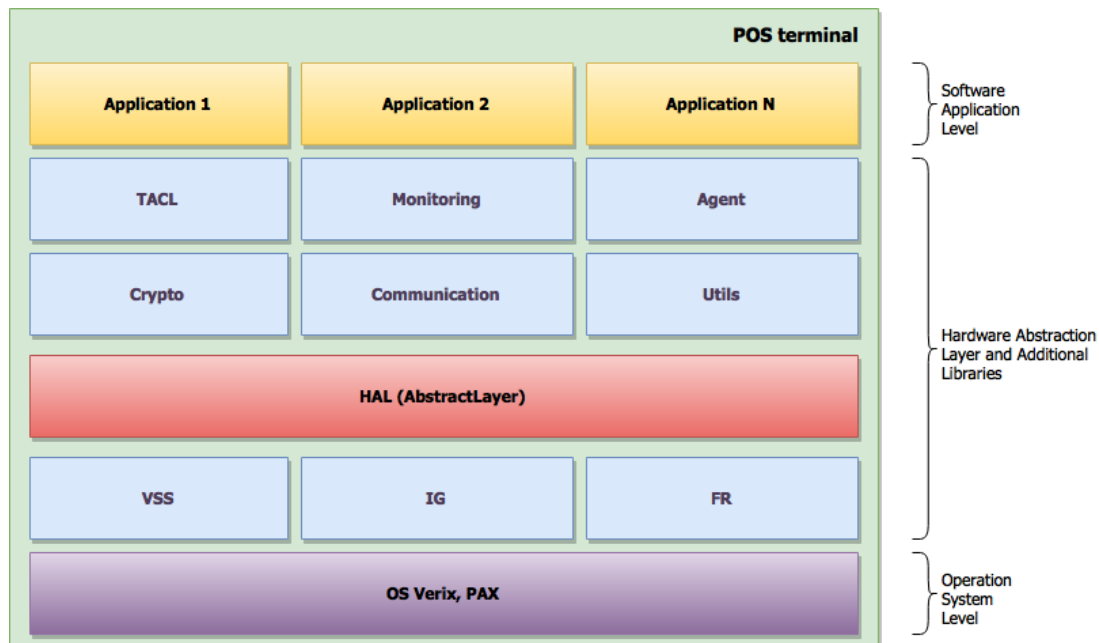


Рисунок 3. Общее расположение и общее описание программного обеспечения POS-терминала

2.1. Crypto Module

Это библиотека, независимая от платформы «Dinamic», которая предназначена для выполнения криптографических функций, таких как шифрование, дешифрование, вычисление MAC-адресов, запрос offline PIN-кода, загрузка ключей в криптомодуль, получение информации о загруженных ключах и преобразование ключей в различные форматы. Существует два варианта реализации: использование PINpad и области сценариев или использование PINpad и модуля SAM.

- **Crypto API:**
 - **Crypto_Encrypt** – эта функция выполняет шифрование с использованием указанного алгоритма:
 - Название – «long Crypto_Encrypt(Crypto_Algorithm algorithm, Crypto_CipherMode mode, const Crypto_Parameters * cryptoParameters, const byte * src, dword sizeSrc, byte * dst)»;
 - Возвращает:
 - SUCCESS — успешное выполнение;
 - ERR_ – список ошибок.
 - Параметры:
 - algorithm [in] – алгоритм шифрования;
 - mode[in] – режим шифрования;
 - cryptoParameters [in] – настройки шифрования;
 - src [in] – входные данные;

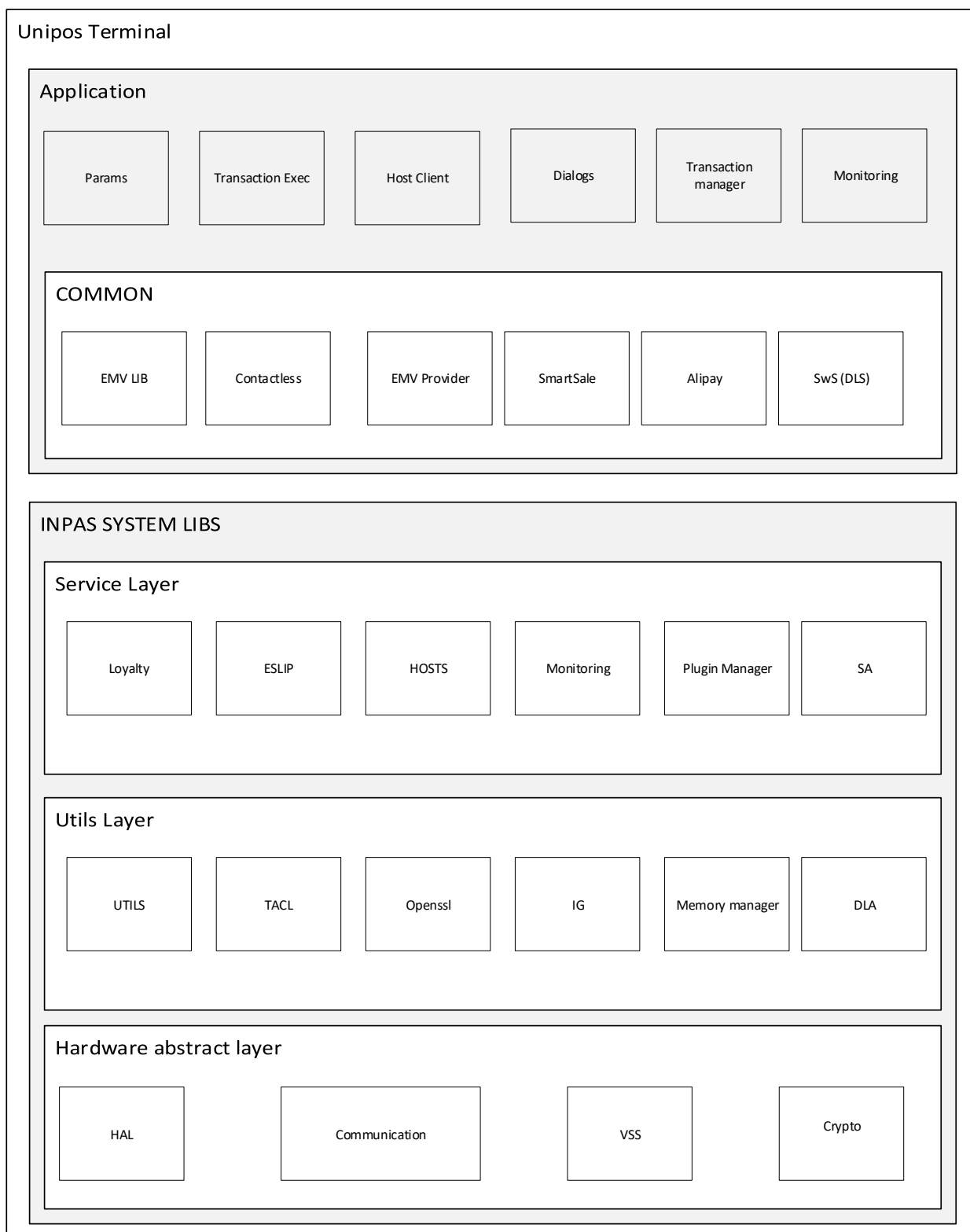
- sizeSrc [in] – размер зашифрованных данных;
 - dst [out] – результат расчета.
- **Crypto_Decrypt** – эта функция выполняет расшифровку с использованием указанного алгоритма:
 - Название – «**long Crypto_Decrypt(Crypto_Algorithm algorithm, Crypto_CipherMode mode, const Crypto_Parameters * cryptoParameters, const byte * src, dword sizeSrc, byte * dst)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_ – список ошибок.
 - Параметры:
 - algorithm [in] – алгоритм шифрования;
 - mode[in] – режим шифрования;
 - cryptoParameters [in] – настройки расшифровки;
 - src [in] – входные данные;
 - sizeSrc [in] – размер дешифрованных данных;
 - dst [out] – результат расчета.
 - **Crypto_CalcMac** – эта функция выполняет расчет MAC с использованием указанного алгоритма:
 - Название – «**long Crypto_CalcMac(Crypto_MacAlgorithm macAlgorithm, const Crypto_Parameters * cryptoParameters, PinpadDataType dataType, const byte * src, dword sizeSrc, char * mac)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_ – список ошибок.
 - Параметры:
 - macAlgorithm [in] – алгоритм для расчета MAC;
 - cryptoParameters [in] – ключи для шифрования данных;
 - dataType [in] – тип входных данных;
 - src [in] – входные данные;
 - sizeSrc [in] – размер входных данных;
 - mac [out] – значение цифровой подписи в ASCII, 16 байт.
 - **Crypto_EnterOfflinePin** – эта функция выполняет запрос offline PIN-кода на PIN-панели, а затем расшифровывает его в криптомодуле:
 - Название – «**long Crypto_EnterOfflinePin(const Crypto_OfflinePinParameters * parameters, char * clearPin)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_PINPAD_USER_CANCEL – клиент отказался вводить PIN-код.
 - Параметры:
 - clearPin [out] – PIN-код в открытом виде.
 - **Crypto_EnterOfflinePinNull** – эта функция выполняет запрос offline PIN-кода на PIN-панели с пустым входным PIN-кодом, а затем расшифровывает его в криптомодуле:
 - Название – «**long Crypto_EnterOfflinePinNull(const Crypto_OfflinePinParameters * parameters, char * clearPin)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_PINPAD_USER_CANCEL – клиент отказался вводить PIN-код.

- Параметры:
 - parameters [in] – запрос параметров offline PIN-кода;
 - clearPin [out] – PIN-код в открытом виде.
- **Crypto_LoadKey** – эта функция загружает ключ в криптомодуль. Если ключ не существует, он будет создан. Если ключ существует, он будет перезаписан:
 - Название – «**long Crypto_LoadKey(const Crypto_KeyInfo * keyInfo, const char * key)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_ – ошибки.
 - Параметры:
 - keyInfo [in] – параметры загруженного ключа:
 - keyInfo->KeyName – номер ключа;
 - keyInfo->Role = KeyRole_LoadEncryptKey – загрузка ключа KLK;
 - keyInfo->Role = KeyRole_Pin – загрузка PIN-кода;
 - keyInfo->Role = KeyRole_Mac – загрузка MAC ключа;
 - keyInfo->Role = KeyRole_OfflinePin – загрузка offline PIN-кода;
 - keyInfo->Role = KeyRole_Crypt – загрузка ключа шифрования / дешифрования;
 - key [in] – значение ключа (32, 120 или 52 for DUKPT 20 ksn + 32 ipek).
- **Crypto_DeleteAllKeys** – эта функция удаляет все ключи загруженные в криптомодуль:
 - Название – «**long Crypto_DeleteAllKeys(void)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_ – ошибки.
- **Crypto_DeletePinpadKeys** – эта функция удаляет все ключи загруженные PINPAD:
 - Название – «**long Crypto_DeletePinpadKeys(void)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_ – ошибки.
- **Crypto_GetKeyInfo** – эта функция получает информацию о ключе, загруженном в криптомодуль:
 - Название – «**long Crypto_GetKeyInfo(Crypto_KeyInfo * keyInfo)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_PINPAD_USER_CANCEL – клиент отказался вводить PIN-код.
 - Параметры:
 - keyInfo [in] – информация о ключе.
- **Crypto_TransformKeyFormat** – эта функция преобразует ключ из формата «ATM» в формат «GISKE»:
 - Название – «**long Crypto_TransformKeyFormat(Crypto_KeyInfo * keyInfo, const char * key, char * transformedKey)**»;
 - Возвращает:
 - SUCCESS – успешное выполнение;
 - ERR_PINPAD_USER_CANCEL – клиент отказался вводить PIN-код.

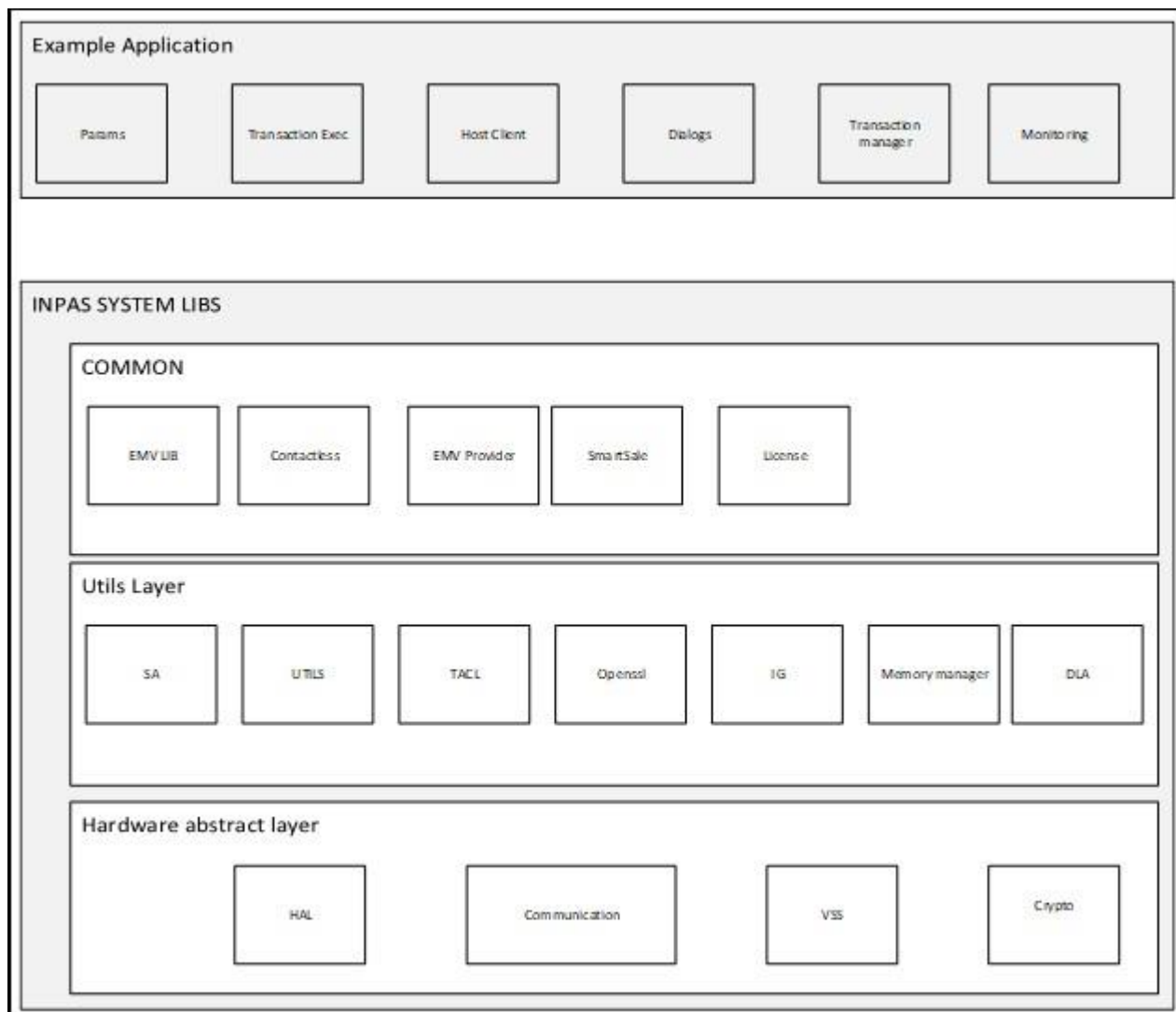
- Параметры:
 - keyInfo [in] – информация о ключе
 - key [in] – значение рабочего ключа в формате «ATM»;
 - transformedKey [out] – рабочий ключ в формате «GISKE».
- **Crypto_TransformKeyToATMFormat** – эта функция преобразует ключ из формата «GISKE» в формат «ATM»:
 - Название – «**long Crypto_TransformKeyToATMFormat(Crypto_KeyInfo * keyInfo, const char * key, char * transformedKey)**»;
 - Возвращает:
 - SUCCESS — успешное выполнение;
 - ERR_PINPAD_USER_CANCEL – клиент отказался вводить PIN-код.
 - Параметры:
 - keyInfo [in] – информация о ключе
 - key [in] – значение рабочего ключа в формате «GISKE»;
 - transformedKey [out] – рабочий ключ в формате «ATM».
- **Crypto_Init** – эта функция инициализирует криптомодуль:
 - Название – «**void Crypto_Init(PinpadType pinpadType, PinpadNumber pinpadNum)**»;
 - Параметры:
 - pinpadType [in] – тип PINPAD’а, который использовался для ввода PIN-кода.

3. Приложения

3.1. Компонентная модель ПО «Unipos»



3.2. Компонентная модель системных библиотек «Unipos»



3.3. Список заголовочных файлов

AdditionalVariables.h	halS200ClssCommands.h
appManager.h	halS200Common.h
appManager2.h	halS200IO.h
coCommon.h	halS200MulappCommands.h
coCommonErrors.h	halS200PedCommands.h
coMacro.h	halS200ScrCommands.h
comCommunication.h	halSHA256.h
comCommunicationParameters.h	halStdlib.h
coTypes.h	halSystem.h
cryCrypto.h	halTerminal.h
dla.h	halTimer.h
DString.h	halTrace.h
halAbstractLayer.h	igBase.h
halBarcode.h	igUI.h
halBattery.h	kkt.h
halBeeper.h	MdbPort.h
halBmp.h	memMemory.h
halChipCardReader.h	PublicEMVKeys.h
halCipher.h	saSess.h
halComPort.h	srvService.h
halContactlessDevice.h	StorageEx.h
halConverter.h	tacl.h
halCrcCalc.h	taclConverter.h
halCryptoAlg.h	taclDataBase.h
halDateTime.h	taclExpressinParser.h
halDesCalc.h	taclEzXml.h
halDir.h	taclIsoMsg.h
halDisplay.h	taclLanguage.h
halDisplayBuf.h	taclLicensing.h
halEmuPrinter.h	taclPrintTemplate.h
halEmuProxy.h	taclQrEncode.h
halExtern.h	taclSet.h
halExternalVssPinPad.h	taclSpdhHeader.h
halFile.h	taclSpdhMsg.h
halFlashDrive.h	taclSpdhTrns.h
halHash.h	taclSpdhTrnsManager.h
halInt64.h	taclSpdhTypes.h
halKeyboard.h	taclXMLParserInterface.h
halLang.h	utiBerTlv.h
halMagCardReader.h	utiBorder.h
halMD5.h	utiCaption.h
halMemory.h	utiCodePage.h
halPinPad.h	utiCodePageManager.h
halPipe.h	utiDialogs.h
halPower.h	utiEdit.h
halPrinter.h	utiFooter.h
halRemoteDevices.h	utilImage.h
halS200BaseCommands.h	utilNi.h
halS200CL_common.h	utiLabel.h

utiList.h
utiListBox.h
utiMonitoring.h
utiResource.h
utiStorage.h
utiText.h
utiTLVParser.h
utiTraceEx.h
utiUtils.h
VmacEvtID.h

vpl.h
vss.h
vssOptCode.h
vssSAPIS.h
wrt.h
xlsconst.h
zlib_iface.h
EmvKernel.h
EmvKernelDef.h