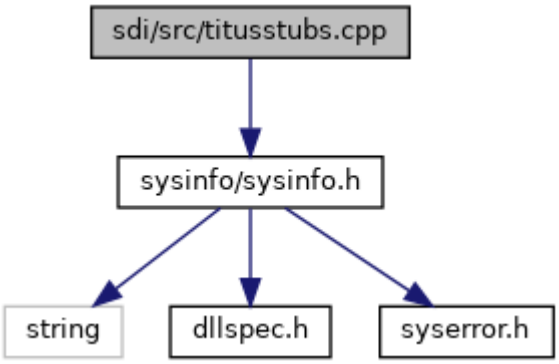
titusstubs.cpp File Reference

```
#include <sysinfo/sysinfo.h>
```

Include dependency graph for titusstubs.cpp:



Data Structures

struct	rawData
struct	discoveryParams
struct	activateParams
struct	discoveryResult
struct	_cardInfo
struct	_pollRes
struct	_cardInfoFull
struct	_pollResFull
struct	_pollReq
struct	authenticationResult
struct	getServicesResult
struct	authenticationParams
struct	getServicesParams
struct	doVASInput
struct	doVASOutput
struct	apduTxData
struct	apduRxData
struct	apduCommand
struct	felicaTxData

struct felicaRxData
struct felicaPolling
struct felicaPollingOutput
struct TX_RX_PARAM
struct callbackFlags
struct callbackText
struct callbackLeds
struct callbackBuzzer
struct callbackInfo

Namespaces

vfigui

Macros

#define UNKNOWN 0x00000000
#define MIFARE_NO_DESFIRE 0x00000001
#define ULTRALIGHT 0x00000002
#define MINI 0x00000004
#define CLASSIC_1K 0x00000008
#define CLASSIC_4K 0x00000010
#define DESFIRE_CL1 0x00000020
#define DESFIRE_CL2 0x00000040
#define PLUS_2K_SL_1 0x00000080
#define PLUS_4K_SL_1 0x00000100
#define PLUS_2K_SL_2 0x00000200
#define PLUS_4K_SL_2 0x00000400
#define PLUS_2K_SL_3 0x00000800
#define PLUS_4K_SL_3 0x00001000
#define TNP3xxx 0x00002000
#define SMART_MX_1K_EMULATION 0x00004000
#define SMART_MX_4K_EMULATION 0x00008000
#define SMART_MX 0x00010000
#define APDU_COMPLIANT 0x00020000
#define NFC_COMPLIANT 0x00040000
#define ULTRALIGHT_C 0x00080000
#define ACTION_DECRYPT 1
#define ACTION_STORE_WALLET_KEYS 3
#define ACTION_GET_WALLET_KEYS 4
#define ACTION_GET_TOKEN 5
#define ACTION_ENCRYPT_TOKEN 6

#define CALLBACK_MAX_TEXT_SIZE (64)

Typedefs

typedef struct _cardInfo cardInfo
typedef struct _pollRes pollRes
typedef struct _cardInfoFull cardInfoFull
typedef struct _pollResFull pollResFull
typedef struct _pollReq pollReq
typedef void() NfcCallbackFunction(unsigned char *data, size_t dataSizeBytes)

Enumerations

ResponseCodes {

EMB_APP_OK = 0,
EMB_APP_INCORRECT_HEADER,
EMB_APP_UNKNOWN_COMMAND,
EMB_APP_UNKNOWN_SUBCOMMAND,

EMB_APP_COMMAND_NOT_SUPPORTED,
EMB_APP_SUBCOMMAND_NOT_SUPPORTED,
EMB_APP_CRC_ERROR,
EMB_APP_FAILED,

EMB_APP_TIMEOUT,
EMB_APP_UNKNOWN_APP_NAME,
EMB_APP_PARAMETER_NOT_SUPPORTED,
EMB_APP_OTHER_APP_RUNNING,

EMB_APP_AUTO_RUN,
EMB_APP_MULTI_CARDS,
EMB_APP_START_PAYMENT,
EMB_APP_COMM_ERROR,

EMB_APP_DATA_CRC_ERROR,
EMB_APP_INCORRECT_DATA,
EMB_APP_CANCEL_DONE,
EMB_APP_CANCEL_IRRELEVANT,

EMB_APP_CANCEL_NOT_ALLOWED,
EMB_APP_DISCOVERY_CANCELED,
EMB_APP_UNSUPPORTED_CARD,
EMB_APP_SECOND_TAP_FAILED,

EMB_APP_OUT_OF_ORDER_COMMAND,
enum EMB_APP_2ND_AUTHENTICATION_FAILED,
EMB_APP_NFC_GROUP_RET_FAIL,
EMB_APP_INCORRECT_LEN,

EMB_APP_TO_MANY_WALLETS,
EMB_APP_COMMAND_NOT_ALLOWED,
EMB_APP_MISSING_MANDATORY_DATA,
EMB_APP_TWO_SNEP_DEFAULT_WALLET,

EMB_APP_INCORRECT_APP_NAME,
EMB_APP_APPLICATION_DATA_NOT_ALLOWED,
EMB_APP_APPLICATION_NOT_FOUND,

VasStatus {

VAS_OK = 0,
VAS_DO_PAY,
VAS_FAIL,
VAS_ERR_CTLS_DRIVER_CLOSE,

VAS_CANCEL,
VAS_TIME_OUT,
VAS_ERR_CONFIG,
VAS_DUMMY_CALL,

enum

VAS_CANCEL_NOT_ALLOWED,
VAS_CANCEL_IRRELEVANT,
VAS_COMM_ERR,
VAS_INIT_ERROR,

VAS_DO_PAY_DECRYPT_REQ,
VAS_OK_DECRYPT_REQ

}

NFC_CARD_TYPE {

I_ISO14443A = 1,
I_JEWEL = 2,
I_ISO14443B = 3,
I_ISO14443BI = 4,

I_ISO14443B2SR = 5,
I_ISO14443B2CT = 6,
I_DEP = 7,
I_FELICA = 8,

enum I_ISO14443A_MIFARE_MINI = 9,
I_ISO14443A_MIFARE_1K = 10,
I_ISO14443A_MIFARE_4K = 11,
I_ISO14443A_MIFARE_DES = 12,

I_ISO14443A_MIFARE_PLUS = 13,
I_ISO14443A_MIFARE_UL = 14,
I_ISO14443A_MIFARE_UL_C = 15,
I_UNKNOWN_CARD_TYPE = 16,

I_ISO14443A_APDU_MIFARE_1K = 17

}

enum I_MIFARE_CARD_TYPE { I_MIFARE_TYPE_CLASSIC,
I_MIFARE_TYPE_ULTRALIGHT,
I_MIFARE_TYPE_ULTRALIGHT_C,
I_MIFARE_TYPE_CLASSIC_4K
}

	NFC_POLL_PARAM_TECH {
	NFC_POLL_PARAM_TECH_A = 0x01,
	NFC_POLL_PARAM_TECH_B = 0x02,
	NFC_POLL_PARAM_TECH_AB = 0x03,
	NFC_POLL_PARAM_TECH_F_DEP = 0x04,
	NFC_POLL_PARAM_TECH_FELICA = 0x08,
enum	NFC_POLL_PARAM_TECH_F = 0x0C,
	NFC_POLL_PARAM_TECH_AF = 0x05,
	NFC_POLL_PARAM_TECH_BF = 0x06,
	NFC_POLL_PARAM_TECH_ABF = 0x07,
	NFC_POLL_PARAM_CUSTOM = 0x10
	}
	NFC_F_BAUD { INF_BAUD212,
enum	INF_BAUD424
	}
	MIFARE_KEY_TYPE { MIFARE_KEY_TYPE_A = 1,
enum	MIFARE_KEY_TYPE_B
	}
	FrameworkState { IDLE = 1,
enum	BUSY = 2
	}
	callbackBuzzerFrequency { CALLBACK_BUZZER_FREQUENCY_LOW = 0xFFFF,
enum	CALLBACK_BUZZER_FREQUENCY_HIGH = 0xFFFE
	}

Functions

bool	isVclEnabled (bool checkVCLStatus)
bool	isVclTag (unsigned long ulTag)
bool	VCL_DecryptMSR ()
bool	VCL_EncryptEMV ()
ResponseCodes	NFC_Config_Init (void)
ResponseCodes	NFC_Get_Version (rawData *output)
ResponseCodes	NFC_Ping (rawData *output)
ResponseCodes	NFC_Set_Callback_Function (rawData *id, NfcCallbackFunction *callbackFunction)

ResponseCodes NFC_Callback_Test (void)

ResponseCodes NFC_PT_Open (void)

ResponseCodes NFC_PT_Close (void)

ResponseCodes NFC_PT_FieldOn (void)

ResponseCodes NFC_PT_FieldOff (void)

ResponseCodes NFC_PT_Polling (pollReq *in_pull_req, pollRes *out_pull_res)

ResponseCodes NFC_PT_PollingFull (pollReq *in_pull_req, pollResFull *out_pull_res)

void NFC_Free_Poll_Data (pollRes *out_pull_res)

void NFC_Free_Poll_DataFull (pollResFull *out_pull_res)

ResponseCodes NFC_PT_Cancel_Polling (void)

ResponseCodes NFC_PT_Activation (NFC_CARD_TYPE cardType, rawData *rd_activationData)

ResponseCodes NFC_PT_TxRx (NFC_CARD_TYPE cardType, rawData *in_buff, rawData *out_buff)

ResponseCodes NFC_PT_FtechBaud (NFC_F_BAUD baud)

ResponseCodes NFC_PT_Adv_TxRx (TX_RX_PARAM *comParams, rawData *in_buff, rawData *out_buff)

ResponseCodes NFC_Mifare_Authenticate (unsigned char blockNumber, MIFARE_KEY_TYPE keyType, rawData *Key)

ResponseCodes NFC_Mifare_Read (I_MIFARE_CARD_TYPE m_cardType, unsigned int StartBlockNum, unsigned int blockAmount, rawData *out_buff)

ResponseCodes NFC_Mifare_Write (I_MIFARE_CARD_TYPE m_cardType, unsigned int StartBlockNum, unsigned int blockAmount, rawData *in_buff)

ResponseCodes NFC_Mifare_Increment (unsigned int blockNum, int amount)

ResponseCodes NFC_Mifare_Decrement (unsigned int blockNum, int amount)

ResponseCodes NFC_Mifare_Increment_Only (unsigned int blockNum, int amount)

ResponseCodes NFC_Mifare_Decrement_Only (unsigned int blockNum, int amount)

ResponseCodes NFC_Mifare_Transfer (unsigned int blockNum)

ResponseCodes NFC_Mifare_Restore (unsigned int blockNum)

ResponseCodes	NFC_Felica_Exchange (felicaTxData *in_buff, felicaRxData *out_buff)
ResponseCodes	NFC_Felica_Polling (unsigned int pollTimeout, felicaPolling *inData, felicaPollingOutput *outData)
ResponseCodes	NFC_APDU_Exchange (apduTxData *txData, apduRxData *rxData)
ResponseCodes	NFC_Card_Removal (void)
VasStatus	NFC_Terminal_Config (rawData *input, rawData *output)
VasStatus	NFC_TERMINAL_ReadConfig (rawData *id, rawData *output)
VasStatus	NFC_VAS_ReadConfig (rawData *id, rawData *output)
VasStatus	NFC_VAS_Activate (rawData *id, rawData *input, rawData *output)
VasStatus	NFC_VAS_Cancel (void)
VasStatus	NFC_VAS_UpdateConfig (rawData *id, rawData *input, rawData *output)
VasStatus	NFC_VAS_CancelConfig (rawData *id)
VasStatus	NFC_VAS_PreLoad (rawData *id, rawData *input, rawData *output)
VasStatus	NFC_VAS_CancelPreLoad (rawData *id)
VasStatus	NFC_VAS_Decrypt (rawData *id, rawData *input, rawData *output)
VasStatus	NFC_VAS_Action (rawData *id, int action, rawData *inData, rawData *outBuff)
DllSpec int	uiSetLED (int display, unsigned led, bool state)
DllSpec int	uiSetLED (unsigned led, bool state)

Data Structure Documentation

? rawData

struct rawData

Data Fields

unsigned char *	buffer
unsigned int	bufferLen
unsigned int	bufferMaxSize

? discoveryParams

struct discoveryParams

Data Fields

unsigned char	activation
unsigned char	cardType
unsigned char	collision
unsigned char	customPollParams[1]

unsigned char customPollParamsSize
unsigned int pollTime
unsigned char technology

? activateParams

struct activateParams

Data Fields

unsigned char technology

? discoveryResult

struct discoveryResult

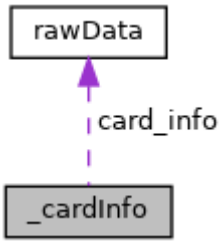
Data Fields

unsigned char cardTypeA
unsigned char cardTypeB
unsigned char cardTypeF
unsigned char numOfCards
unsigned char numOfCardsA
unsigned char numOfCardsB
unsigned char numOfCardsF
unsigned char pollData[1]
unsigned char pollDataSize
unsigned char sak
unsigned char status
unsigned char uidA[1]
unsigned char uidA_Size
unsigned char uidB[1]
unsigned char uidB_Size
unsigned char uidF[1]
unsigned char uidF_Size

? _cardInfo

struct _cardInfo

Collaboration diagram for _cardInfo:



[legend]

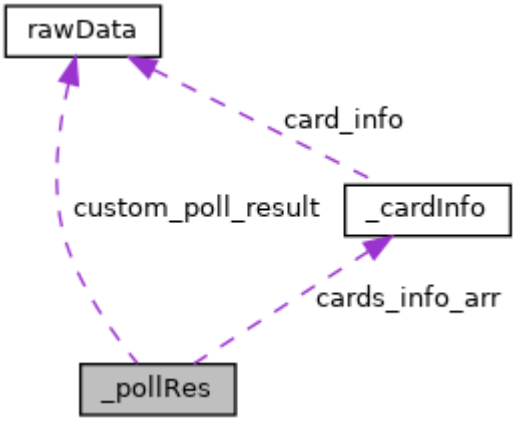
Data Fields

rawData	card_info
NFC_CARD_TYPE	cardType

? _pollRes

struct _pollRes

Collaboration diagram for _pollRes:



[legend]

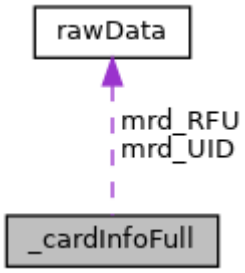
Data Fields

cardInfo *	cards_info_arr
rawData	custom_poll_result
unsigned int	m_foundTargetsA
unsigned int	m_foundTargetsB
unsigned int	m_foundTargetsF
unsigned int	m_foundTargetsTotalCount

? _cardInfoFull

struct _cardInfoFull

Collaboration diagram for _cardInfoFull:



[legend]

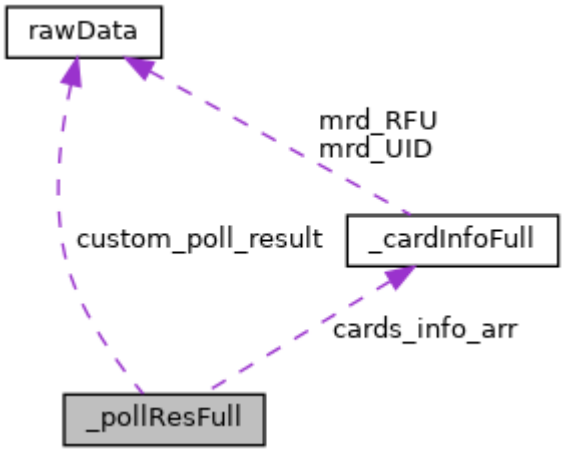
Data Fields

unsigned char	ATQ[2]
long	m_cardTypes
unsigned char	m_modulation
unsigned char	m_SAK
rawData	mrd_RFU
rawData	mrd_UID

? _pollResFull

struct _pollResFull

Collaboration diagram for _pollResFull:



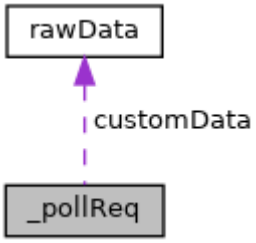
[legend]

Data Fields

cardInfoFull *	cards_info_arr
rawData	custom_poll_result
unsigned int	m_foundTargetsA
unsigned int	m_foundTargetsB
unsigned int	m_foundTargetsF
unsigned int	m_foundTargetsTotalCount

? _pollReq

struct _pollReq
Collaboration diagram for _pollReq:

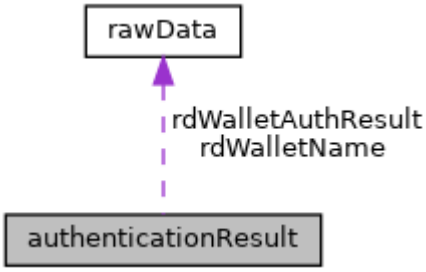


[legend]

Data Fields	
rawData	customData
unsigned int	poll_timeout
unsigned int	tech_bitmap
bool	turnDutyCycleOn

? authenticationResult

struct authenticationResult
Collaboration diagram for authenticationResult:

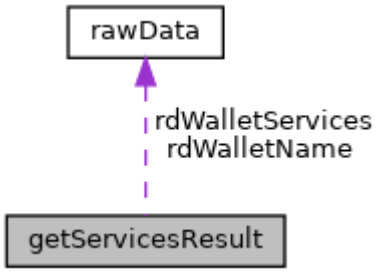


[legend]

Data Fields	
rawData	rdWalletAuthResult
rawData	rdWalletName
unsigned char	status

? getServicesResult

struct getServicesResult
Collaboration diagram for getServicesResult:



[legend]

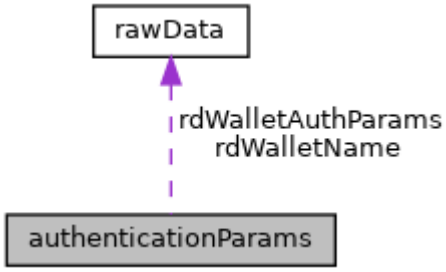
Data Fields

rawData	rdWalletName
rawData	rdWalletServices
unsigned char	status

? authenticationParams

struct authenticationParams

Collaboration diagram for authenticationParams:



[legend]

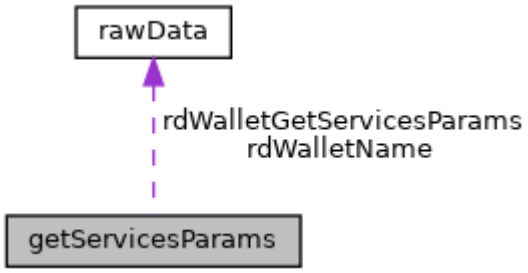
Data Fields

unsigned char	cardType
unsigned int	expectedResultMaxSize
rawData	rdWalletAuthParams
rawData	rdWalletName
unsigned char	tech

? getServicesParams

struct getServicesParams

Collaboration diagram for getServicesParams:



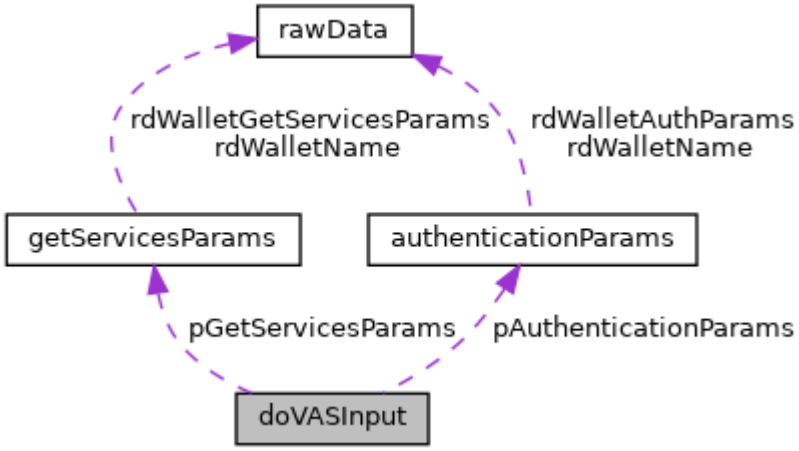
[legend]

Data Fields

unsigned int expectedResultMaxSize
rawData rdWalletGetServicesParams
rawData rdWalletName

? doVASInput

struct doVASInput
Collaboration diagram for doVASInput:



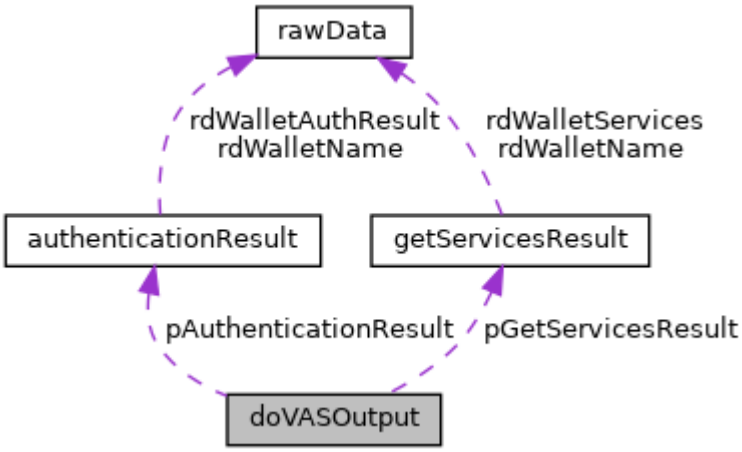
[legend]

Data Fields

authenticationParams * pAuthenticationParams
getServicesParams * pGetServicesParams

? doVASOutput

struct doVASOutput
Collaboration diagram for doVASOutput:



[legend]

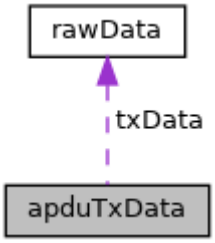
Data Fields

authenticationResult *	pAuthenticationResult
getServicesResult *	pGetServicesResult

? apduTxData

struct apduTxData

Collaboration diagram for apduTxData:



[legend]

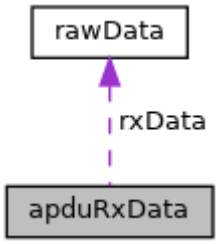
Data Fields

unsigned int	clas
unsigned int	expectedResponseLen
unsigned int	instruction
unsigned int	param1
unsigned int	param2
rawData	txData

? apduRxData

struct apduRxData

Collaboration diagram for apduRxData:



[legend]

Data Fields

rawData rxData
unsigned int sw1
unsigned int sw2

? apduCommand

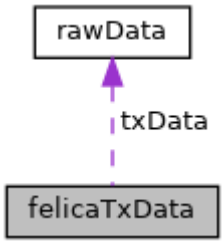
struct apduCommand

Data Fields

unsigned char clas
unsigned char expectResLen
unsigned char instraction
unsigned char param1
unsigned char param2

? felicaTxData

struct felicaTxData
Collaboration diagram for felicaTxData:



[legend]

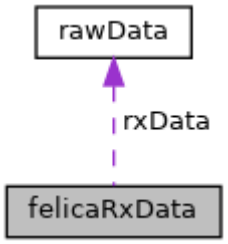
Data Fields

unsigned int timeOut
rawData txData

? felicaRxData

struct felicaRxData

Collaboration diagram for felicaRxData:



[legend]

Data Fields

rawData rxData

? felicaPolling

struct felicaPolling

Data Fields

unsigned int reciveTimeOUt
unsigned char requestCode
unsigned char systemCode[2]
unsigned char TimeSlot

? felicaPollingOutput

struct felicaPollingOutput

Data Fields

unsigned char pad[8]
unsigned char sysCode[2]
unsigned char UID[8]

? TX_RX_PARAM

struct TX_RX_PARAM

Data Fields

unsigned char CRC
unsigned char invalidFrames
unsigned char parity
unsigned int responseTimeout
unsigned char tech
unsigned int timeOutCom

? callbackFlags

struct callbackFlags

Data Fields

unsigned int buzzerPresent: 1
unsigned int init: 1
unsigned int ledsPresent: 1
unsigned int textPresent: 1

? callbackText

struct callbackText

Data Fields

char text[CALLBACK_MAX_TEXT_SIZE]

? callbackLeds

struct callbackLeds

Data Fields

unsigned int led1: 2
unsigned int led2: 2
unsigned int led3: 2
unsigned int led4: 2

? callbackBuzzer

struct callbackBuzzer

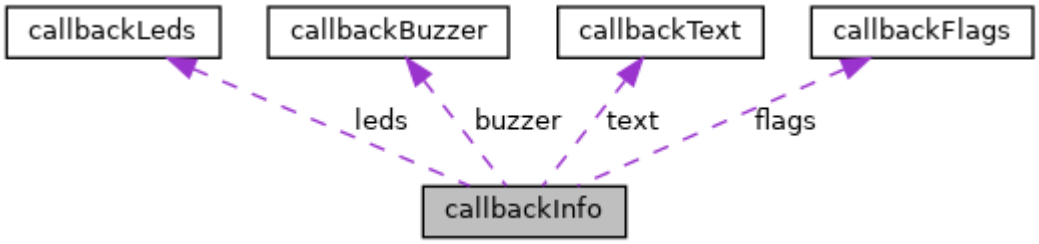
Data Fields

callbackBuzzerFrequency	frequency
unsigned int	nTimes
unsigned int	off_ms
unsigned int	on_ms

? callbackInfo

struct callbackInfo

Collaboration diagram for callbackInfo:



[legend]

Data Fields

callbackBuzzer	buzzer
callbackFlags	flags
callbackLeds	leds
callbackText	text

Macro Definition Documentation

? ACTION_DECRYPT

#define ACTION_DECRYPT 1

? ACTION_ENCRYPT_TOKEN

#define ACTION_ENCRYPT_TOKEN 6

? ACTION_GET_TOKEN

#define ACTION_GET_TOKEN 5

? ACTION_GET_WALLET_KEYS

#define ACTION_GET_WALLET_KEYS 4

? ACTION_STORE_WALLET_KEYS

#define ACTION_STORE_WALLET_KEYS 3

? APDU_COMPLIANT

#define APDU_COMPLIANT 0x00020000

? CALLBACK_MAX_TEXT_SIZE

#define CALLBACK_MAX_TEXT_SIZE (64)

? CLASSIC_1K

#define CLASSIC_1K 0x00000008

? CLASSIC_4K

#define CLASSIC_4K 0x00000010

? DESFIRE_CL1

#define DESFIRE_CL1 0x00000020

? DESFIRE_CL2

#define DESFIRE_CL2 0x00000040

? MIFARE_NO_DESFIRE

#define MIFARE_NO_DESFIRE 0x00000001

? MINI

#define MINI 0x00000004

? NFC_COMPLIANT

#define NFC_COMPLIANT 0x00040000

? PLUS_2K_SL_1

#define PLUS_2K_SL_1 0x00000080

? PLUS_2K_SL_2

#define PLUS_2K_SL_2 0x00000200

? PLUS_2K_SL_3

#define PLUS_2K_SL_3 0x00000800

? PLUS_4K_SL_1

#define PLUS_4K_SL_1 0x00000100

? PLUS_4K_SL_2

#define PLUS_4K_SL_2 0x00000400

? PLUS_4K_SL_3

#define PLUS_4K_SL_3 0x00001000

? SMART_MX

#define SMART_MX 0x00010000

? SMART_MX_1K_EMULATION

#define SMART_MX_1K_EMULATION 0x00004000

? SMART_MX_4K_EMULATION

#define SMART_MX_4K_EMULATION 0x00008000

? TNP3xxx

#define TNP3xxx 0x00002000

? ULTRALIGHT

#define ULTRALIGHT 0x00000002

? ULTRALIGHT_C

#define ULTRALIGHT_C 0x00080000

? UNKNOWN

#define UNKNOWN 0x00000000

Typedef Documentation

? cardInfo

typedef struct _cardInfo cardInfo

? cardInfoFull

typedef struct _cardInfoFull cardInfoFull

? NfcCallbackFunction

typedef void() NfcCallbackFunction(unsigned char *data, size_t dataSizeBytes)

? pollReq

typedef struct _pollReq pollReq

? pollRes

typedef struct _pollRes pollRes

? pollResFull

typedef struct _pollResFull pollResFull

Enumeration Type Documentation

? callbackBuzzerFrequency

enum callbackBuzzerFrequency

Enumerator
CALLBACK_BUZZER_FREQUENCY_LOW
CALLBACK_BUZZER_FREQUENCY_HIGH

? FrameworkState

enum FrameworkState

Enumerator
IDLE
BUSY

? I_MIFARE_CARD_TYPE

enum I_MIFARE_CARD_TYPE

Enumerator
I_MIFARE_TYPE_CLASSIC
I_MIFARE_TYPE_ULTRALIGHT
I_MIFARE_TYPE_ULTRALIGHT_C
I_MIFARE_TYPE_CLASSIC_4K

? MIFARE_KEY_TYPE

enum MIFARE_KEY_TYPE

Enumerator
MIFARE_KEY_TYPE_A
MIFARE_KEY_TYPE_B

? NFC_CARD_TYPE

enum NFC_CARD_TYPE

Enumerator
I_ISO14443A
I_JEWEL
I_ISO14443B
I_ISO14443BI

I_ISO14443B2SR
I_ISO14443B2CT
I_DEP
I_FELICA
I_ISO14443A_MIFARE_MINI
I_ISO14443A_MIFARE_1K
I_ISO14443A_MIFARE_4K
I_ISO14443A_MIFARE_DES
I_ISO14443A_MIFARE_PLUS
I_ISO14443A_MIFARE_UL
I_ISO14443A_MIFARE_UL_C
I_UNKNOWN_CARD_TYPE
I_ISO14443A_APDU_MIFARE_1K

? NFC_F_BAUD

enum NFC_F_BAUD

Enumerator
INF_BAUD212
INF_BAUD424

? NFC_POLL_PARAM_TECH

enum NFC_POLL_PARAM_TECH

Enumerator
NFC_POLL_PARAM_TECH_A
NFC_POLL_PARAM_TECH_B
NFC_POLL_PARAM_TECH_AB
NFC_POLL_PARAM_TECH_F_DEP
NFC_POLL_PARAM_TECH_FELICA
NFC_POLL_PARAM_TECH_F
NFC_POLL_PARAM_TECH_AF
NFC_POLL_PARAM_TECH_BF
NFC_POLL_PARAM_TECH_ABF
NFC_POLL_PARAM_CUSTOM

? ResponseCodes

enum ResponseCodes

Enumerator

EMB_APP_OK	EMB_APP_OK.
EMB_APP_INCORRECT_HEADER	
EMB_APP_UNKNOWN_COMMAND	
EMB_APP_UNKNOWN_SUBCOMMAND	
EMB_APP_COMMAND_NOT_SUPPORTED	
EMB_APP_SUBCOMMAND_NOT_SUPPORTED	
EMB_APP_CRC_ERROR	
EMB_APP_FAILED	
EMB_APP_TIMEOUT	
EMB_APP_UNKNOWN_APP_NAME	
EMB_APP_PARAMETER_NOT_SUPPORTED	
EMB_APP_OTHER_APP_RUNNING	
EMB_APP_AUTO_RUN	
EMB_APP_MULTI_CARDS	
EMB_APP_START_PAYMENT	
EMB_APP_COMM_ERROR	
EMB_APP_DATA_CRC_ERROR	
EMB_APP_INCORRECT_DATA	
EMB_APP_CANCEL_DONE	
EMB_APP_CANCEL_IRRELEVANT	
EMB_APP_CANCEL_NOT_ALLOWED	
EMB_APP_DISCOVERY_CANCELED	
EMB_APP_UNSUPPORTED_CARD	
EMB_APP_SECOND_TAP_FAILED	
EMB_APP_OUT_OF_ORDER_COMMAND	
EMB_APP_2ND_AUTHENTICATION_FAILED	
EMB_APP_NFC_GROUP_RET_FAIL	
EMB_APP_INCORRECT_LEN	
EMB_APP_TO_MANY_WALLETS	
EMB_APP_COMMAND_NOT_ALLOWED	
EMB_APP_MISSING_MANDATORY_DATA	
EMB_APP_TWO_SNEP_DEFAULT_WALLET	
EMB_APP_INCORRECT_APP_NAME	
EMB_APP_APPLICATION_DATA_NOT_ALLOWED	
EMB_APP_APPLICATION_NOT_FOUND	
EMB_APP_MISMATHCED_UID	
EMB_APP_WALLET_NOT_EXIST	
EMB_APP_NFCUTIL_ERROR	

EMB_APP_USER_ACTION_REQUIRED_UI
EMB_APP_VAS_DATA_NOT_ACTIVATED
EMB_APP_BAD_SYNTAX
EMB_APP_INTERNAL_ERROR
EMB_APP_CARD_NOT_FOUND
EMB_APP_CARD_ERROR
EMB_APP_L1_CARD_PROTOCOL_ERROR
EMB_APP_L1_CARD_NOT_ACTIVE
EMB_APP_DUMMY_FUNC_CALL
EMB_APP_CALLBACK_SET_ERROR
EMB_APP_INIT_ERROR
EMB_APP_L1_DRIVER_CLOSED

? VasStatus

enum VasStatus

Enumerator
VAS_OK
VAS_DO_PAY
VAS_FAIL
VAS_ERR_CTLS_DRIVER_CLOSE
VAS_CANCEL
VAS_TIME_OUT
VAS_ERR_CONFIG
VAS_DUMMY_CALL
VAS_CANCEL_NOT_ALLOWED
VAS_CANCEL_IRRELEVANT
VAS_COMM_ERR
VAS_INIT_ERROR
VAS_DO_PAY_DECRYPT_REQ
VAS_OK_DECRYPT_REQ

Function Documentation

? isVclEnabled()

bool isVclEnabled (bool *checkVCLStatus*)

? isVclTag()

bool isVclTag (unsigned long *ulTag*)

? NFC_APDU_Exchange()

ResponseCodes NFC_APDU_Exchange (apduTxData * *txData*,
apduRxData * *rxData*
)

Data transceive of APDU protocol.

This command is not allowed on SDI Server external interface

Parameters

[in] txData apduTxData data to send

[out] rxData apduRxData data received

Returns

EMB_APP_COMMAND_NOT_SUPPORTED

? NFC_Callback_Test()

ResponseCodes NFC_Callback_Test (void)

Undocumented function, just included because part of NFC_Interface.h.

Returns

EMB_APP_COMMAND_NOT_SUPPORTED

? NFC_Card_Removal()

ResponseCodes NFC_Card_Removal (void)

? NFC_Config_Init()

ResponseCodes NFC_Config_Init (void)

According NFC documentation: Initializes NFC Configuration. But this function does nothing and always returns error.

Returns
EMB_APP_FAILED

? NFC_Felica_Exchange()

ResponseCodes NFC_Felica_Exchange (felicaTxData * *in_buff*,
felicaRxData * *out_buff*
)

Data transive over Felica protocol

Parameters
[in] in_buff binary input
[out] out_buff binary output

Returns
ResponseCodes

? NFC_Felica_Polling()

ResponseCodes NFC_Felica_Polling (unsigned int *pollTimeout*,
felicaPolling * *inData*,
felicaPollingOutput * *outData*
)

FeliCa Polling request

Parameters
[in] pollTimeout timeout in Milli Seconds
[in] inData felicaPolling input data
[out] outData felicaPollingOutput result data

Returns
ResponseCodes

? NFC_Free_Poll_Data()

void NFC_Free_Poll_Data (pollRes * *outPollRes*)

Releases memory allocated in the pollRes *outPollRes when NFC_PT_Polling() was called.

? NFC_Free_Poll_DataFull()

void NFC_Free_Poll_DataFull (pollResFull * *out_pull_res*)

? NFC_Get_Version()

ResponseCodes NFC_Get_Version (rawData * *output*)

Returns ADK-NFC build and kernels versions as JSON string Depending on return code getNfcSW12() or getNfcClientError() might provide the error reason.

Parameters

[out] output data buffer for the version information

Returns

NFC result, EMB_APP_FAILED for other SW12, EMB_APP_COMM_ERROR for client side error

? NFC_Mifare_Authenticate()

ResponseCodes NFC_Mifare_Authenticate (unsigned char *blockNumber*,
MIFARE_KEY_TYPE *keyType*,
rawData * *Key*
)

Authenticates the section with given block address with either key A or B.

Parameters

[in] blockNumber MIFARE block address within the section to authenticate '00' .. 'FF'
[in] keyType MIFARE_KEY_TYPE_A or MIFARE_KEY_TYPE_B
[in] Key MIFARE key (e.g. MIFARE classic crypto-1 key with 48 bit)

Returns

ResponseCodes

? NFC_Mifare_Decrement()

ResponseCodes NFC_Mifare_Decrement (unsigned int *blockNum*,
int *amount*
)

Decrement MIFARE value block by amount and transfer to original location

Parameters

[in] blockNum source and destination block address

[in] amount 4 byte signed integer

Returns

ResponseCodes

? NFC_Mifare_Decrement_Only()

ResponseCodes NFC_Mifare_Decrement_Only (unsigned int *blockNum*,
int *amount*
)

Decrement MIFARE value block by amount and store at transfer buffer

Parameters

[in] blockNum source block address

[in] amount 4 byte signed integer

Returns

ResponseCodes

? NFC_Mifare_Increment()

ResponseCodes NFC_Mifare_Increment (unsigned int *blockNum*,
int *amount*
)

Increment MIFARE value block by amount and transfer to original location

Parameters

[in] blockNum source and destination block address

[in] amount 4 byte signed integer

Returns

ResponseCodes

? NFC_Mifare_Increment_Only()

ResponseCodes NFC_Mifare_Increment_Only (unsigned int *blockNum*,
int *amount*
)

Increment MIFARE value block by amount and store at transfer buffer

Parameters
[in] blockNum source block address
[in] amount 4 byte signed integer

Returns
ResponseCodes

? NFC_Mifare_Read()

ResponseCodes NFC_Mifare_Read (I_MIFARE_CARD_TYPE *m_cardType*,
unsigned int *StartBlockNum*,
unsigned int *blockAmount*,
rawData * *out_buff*
)

Read block data of up to 15 blocks

Parameters
[in] m_cardType I_MIFARE_CARD_TYPE
[in] StartBlockNum address of first block
[in] blockAmount number of blocks to read
[out] out_buff output buffer, required size is 16*blockAmount

Returns
ResponseCodes

? NFC_Mifare_Restore()

ResponseCodes NFC_Mifare_Restore (unsigned int *blockNum*)

Write value from source block to transfer buffer

Parameters
[in] blockNum source block address

Returns

ResponseCodes

? NFC_Mifare_Transfer()

ResponseCodes NFC_Mifare_Transfer (unsigned int *blockNum*)

Write value from transfer buffer to destination block

Parameters
[in] blockNum destination block address

Returns
ResponseCodes

? NFC_Mifare_Write()

ResponseCodes NFC_Mifare_Write (I_MIFARE_CARD_TYPE *m_cardType*,
unsigned int *StartBlockNum*,
unsigned int *blockAmount*,
rawData * *in_buff*
)

Write block data of up to 15 blocks

Parameters

[in] m_cardType	I_MIFARE_CARD_TYPE
[in] StartBlockNum	address of first block
[in] blockAmount	number of blocks to read
[in] in_buff	data to write of size 16*blockAmount

Returns
ResponseCodes

? NFC_Ping()

ResponseCodes NFC_Ping (rawData * *output*)

Return NFC Framework State of the NFC framework.

? NFC_PT_Activation()

ResponseCodes NFC_PT_Activation (NFC_CARD_TYPE *cardtype*,
rawData * *rd_activationData*
)

Activates (selects) the card found during polling. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

Parameters

[in] cardtype	value from pollRes::cardInfo]n].cardType or pollResFull::cards_info_arr [n].m_modulation
[in] rd_activationData	value from pollRes::cards_info_arr[n].card_info or pollResFull::card_info_arr[n].mrd_UID

? NFC_PT_Adv_TxRx()

ResponseCodes NFC_PT_Adv_TxRx (TX_RX_PARAM * *comParams*,
rawData * *in_buff*,
rawData * *out_buff*
)

? NFC_PT_Cancel_Polling()

ResponseCodes NFC_PT_Cancel_Polling (void)

Stop polling before timeout. Note: This command has to be send asynchronously while waiting for polling response. Since is not yet supported on SDI Server side there is also no implementation on client side.

Returns
EMB_APP_COMMAND_NOT_SUPPORTED

? NFC_PT_Close()

ResponseCodes NFC_PT_Close ( )

Release NFC L1 driver. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

? NFC_PT_FieldOff()

ResponseCodes NFC_PT_FieldOff ( )

Turns RF field off. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

? NFC_PT_FieldOn()

ResponseCodes NFC_PT_FieldOn ( )

Turns RF field on. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

? NFC_PT_FtechBaud()

ResponseCodes NFC_PT_FtechBaud (NFC_F_BAUD *baud*)

Changes NFC-F baud rate

Parameters

[in] baud value of NFC_F_BAUD

Returns

value of ResponseCodes

? NFC_PT_Open()

ResponseCodes NFC_PT_Open ( )

Initialise NFC L1 driver. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

? NFC_PT_Polling()

ResponseCodes NFC_PT_Polling (pollReq * *inPollReq*,
pollReq * *outPollRes*
)

Activates polling. See NFC documentation. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

? NFC_PT_PollingFull()

```
ResponseCodes NFC_PT_PollingFull ( pollReq *      inPollReq,
                                   pollResFull *  outPollRes
                                 )
```

Activates polling. See NFC documentation. When the SDI server response is not 90xx or there was a communication problem an appropriate CL_STATUS or CL_STATUS_GENERAL_ERROR is returned. The functions getNfcClientError and getNfcSW12 will provide the error indication.

? NFC_PT_TxRx()

```
ResponseCodes NFC_PT_TxRx ( NFC_CARD_TYPE  cardtype,
                            rawData *      inBuff,
                            rawData *      outBuff
                          )
```

Sends and receives raw data using ISO 14443-3 protocol (31-08) This function is no more available at SDI server external interface

Returns
EMB_APP_COMMAND_NOT_SUPPORTED

? NFC_Set_Callback_Function()

```
ResponseCodes NFC_Set_Callback_Function ( rawData *               id,
                                          NfcCallbackFunction *   callbackFunction
                                        )
```

Set UI callback function handling text, status indicators (LEDs) and buzzer. So far, there is no client side implementation.

Returns
EMB_APP_COMMAND_NOT_SUPPORTED

? NFC_Terminal_Config()

VasStatus NFC_Terminal_Config (rawData * *input*,
rawData * *output*
)

Terminal wide VAS configuration

Parameters
[in] input JSON string see Terminal Configuration Parameters
[out] output nothing - RFU

Returns
VasStatus

? NFC_TERMINAL_ReadConfig()

VasStatus NFC_TERMINAL_ReadConfig (rawData * *id*,
rawData * *output*
)

Reads the most updated terminal configuration.Static parameter will be returned in case appID is unknown or NFC_VAS_PreLoad() issued without changing Terminal configuration.

Parameters
[in] id application unique identifier
[out] output JSON string most updated Read Terminal Configuration Parameters

Returns
VasStatus

? NFC_VAS_Action()

VasStatus NFC_VAS_Action (rawData * *id*,
int *action*,
rawData * *inData*,
rawData * *outBuff*
)

Key transfer between Counter Top and External PIN pad.

Not to be used at external SDI Server interface

Returns
VasStatus

? NFC_VAS_Activate()

VasStatus NFC_VAS_Activate (rawData * *id*,
rawData * *input*,
rawData * *output*
)

Activates NFC interface, runs through wallet kernel flow and retrieves VAS data.

Parameters
[in] id application unique identifier
[in] input JSON string set of dynamic parameters to be merged with configuration from data base.
[out] output JSON string VAS data received from the mobile.

Returns
VasStatus

? NFC_VAS_Cancel()

VasStatus NFC_VAS_Cancel (void)

Stop VAS activate polling before timeout. Note: This command has to be send asynchronously while waiting for polling response. Since is not yet supported on SDI Server side there is also no implementation on client side.

Returns
EMB_APP_COMMAND_NOT_SUPPORTED

? NFC_VAS_CancelConfig()

VasStatus NFC_VAS_CancelConfig (rawData * *id*)

Clears all the VAS configuration by application ID

Parameters
[in] id application unique identifier

Returns
VasStatus

? NFC_VAS_CancelPreLoad()

VasStatus NFC_VAS_CancelPreLoad (rawData * *id*)

Clear preloaded configuration by application ID and pulls latest static configuration from data base.

Parameters
[in] id application unique identifier

Returns
VasStatus

? NFC_VAS_Decrypt()

VasStatus NFC_VAS_Decrypt (rawData * *id*,
rawData * *input*,
rawData * *output*
)

Decrypts an encrypted VAS response.

Parameters
[in] id application unique identifier
[in] input The json in the same format of the Vas Data Response with the included encrypted message
[out] output The json in the same format of the Vas Data Response with the included decrypted message

Returns
VasStatus

? NFC_VAS_PreLoad()

VasStatus NFC_VAS_PreLoad (rawData * *id*,
rawData * *input*,
rawData * *output*
)

Configures the terminal with wallet specific parameters. NFC_VAS_Activate() has to be called to get VAS data. Only single PreLoaded configuration is available.

Parameters

[in] id application unique identifier

[in] input input Set of PreLoad parameters to be merged with configuration from data base.

[out] output none - RFU

Returns

VasStatus

? NFC_VAS_ReadConfig()

```
VasStatus NFC_VAS_ReadConfig ( rawData *  id,
                               rawData *  output
                             )
```

Reads the most updated wallets configuration.

Parameters

[in] id application unique identifier

[out] output JSON string most updated configuration for terminal and all wallets

Returns

VasStatus

? NFC_VAS_UpdateConfig()

```
VasStatus NFC_VAS_UpdateConfig ( rawData *  id,
                                 rawData *  input,
                                 rawData *  output
                               )
```

Configures the terminal with wallet specific parameters.

Parameters

[in] id application unique identifier

[in] input JSON string set of parameters to configure ome or multiple wallets

[out] output none - RFU

Returns

VasStatus

? VCL_DecryptMSR()

bool VCL_DecryptMSR ( )

? VCL_EncryptEMV()

bool VCL_EncryptEMV ( )