Computer Network Protocol (CNP) Project

Table of Contents

Main Page

Author:
Mark L. Short

Date:
March 25, 2015

Course :

5580 Computer Networks

Objective:

Objective of this group activity is to design an application layer protocol standard that will be used by all groups for the project. The protocol must provide the basic ATM Banking functionality of:

- Establish a connection
- Creating an Account
  - First Name
  - Last Name
  - Email Address
  - SSN
  - Driver's License #
  - User selected PIN
- Logging On
  - PIN & First Name
- Deposit
  - Cash or Check
- Withdrawal
- Stamp Purchase
- Transaction History Query

Implementation Notes

1. In addition to those required functions, the following were implemented:
   - Logging Off (explicit)
   - Balance Query
2. Those types with the prefixed '_' are intentionally 'uglified' to discourage their direct use. Additionally, they have been wrapped in the 'prim' namespace to further obscure them from direct use.

This is a common naming procedure used to denote an 'internal' type within a published API specification or protocol to not directly use.

3. Regarding inheritance.

Even though, from an implementation stand-point, C++ inheritance would have afforded the ability to avoid a lot of "helper" method duplication; from the position of affording any byte-wise guarantee of consistency across various compilers and platforms, there was no way to insure that the use of inheritance would not, covertly, introduce additional hidden bytes or other compiler generated information.

So, it was explicitly avoided in the message protocol implementation.

Module Index

Modules

Here is a list of all modules:

Namespace Index

Namespace List

Here is a list of all namespaces with brief descriptions:

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

File List

Here is a list of all files with brief descriptions:

Module Documentation

CNP Protocol Messages

Modules

- Server Messages
- Client Messages
- Message TypeDefs

Detailed Description

Server Messages

Classes

- struct cnp::CONNECT_RESPONSE
- *[Server] Connect Response message* struct cnp::CREATE_ACCOUNT_RESPONSE
- *[Server] Create Account Response message* struct cnp::LOGON_RESPONSE
- *[Server] Logon Response message* struct cnp::LOGOFF_RESPONSE
- *[Server] Logoff Response message* struct cnp::DEPOSIT_RESPONSE

- *[Server] Deposit Response message* struct cnp::WITHDRAWAL_RESPONSE
- *[Server] Withdrawal Response message* struct cnp::BALANCE_QUERY_RESPONSE
- *[Server] Balance Query Response message* struct cnp::TRANSACTION_QUERY_RESPONSE
- *[Server] Transaction Query Response message* struct cnp::STAMP_PURCHASE_RESPONSE

[Server] Stamp Purchase Response message

Detailed Description

Client Messages

Classes

- struct cnp::CONNECT_REQUEST
- *[Client] Connect Request message* struct cnp::CREATE_ACCOUNT_REQUEST
- *[Client] Create Account Request message* struct cnp::LOGON_REQUEST
- *[Client] Logon Request message* struct cnp::LOGOFF_REQUEST
- *[Client] Logoff Request message* struct cnp::DEPOSIT_REQUEST
- *[Client] Deposit Request message* struct cnp::WITHDRAWAL_REQUEST
- *[Client] Withdrawal Request message* struct cnp::BALANCE_QUERY_REQUEST
- *[Client] Balance Query Request message* struct cnp::TRANSACTION_QUERY_REQUEST
- *[Client] Transaction Query Request message* struct cnp::STAMP_PURCHASE_REQUEST

[Client] Stamp Purchase Request Message

Detailed Description

Message TypeDefs

Classes

- struct cnp::TRANSACTION
- *A Customer Transaction Record.* struct cnp::STD_HDR

CNP Standard Message Header. Enumerations

- enum cnp::MSG_TYPE { cnp::MT_INVALID = MAKE_MSG_TYPE(CMT_INVALID, CMS_INVALID), cnp::MT_CONNECT_REQUEST = MAKE_MSG_TYPE(CMT_CONNECT, CMS_REQUEST), cnp::MT_CONNECT_RESPONSE = MAKE_MSG_TYPE(CMT_CONNECT, CMS_RESPONSE), cnp::MT_CREATE_ACCOUNT_REQUEST = MAKE_MSG_TYPE(CMT_CREATE_ACCOUNT, CMS_REQUEST), cnp::MT_CREATE_ACCOUNT_RESPONSE = MAKE_MSG_TYPE(CMT_CREATE_ACCOUNT, CMS_RESPONSE), cnp::MT_LOGON_REQUEST = MAKE_MSG_TYPE(CMT_LOGON, CMS_REQUEST), cnp::MT_LOGON_RESPONSE = MAKE_MSG_TYPE(CMT_LOGON, CMS_RESPONSE), cnp::MT_LOGOFF_REQUEST = MAKE_MSG_TYPE(CMT_LOGOFF, CMS_REQUEST), cnp::MT_LOGOFF_RESPONSE = MAKE_MSG_TYPE(CMT_LOGOFF, CMS_RESPONSE), cnp::MT_DEPOSIT_REQUEST = MAKE_MSG_TYPE(CMT_DEPOSIT, CMS_REQUEST), cnp::MT_DEPOSIT_RESPONSE = MAKE_MSG_TYPE(CMT_DEPOSIT, CMS_RESPONSE), cnp::MT_WITHDRAWAL_REQUEST = MAKE_MSG_TYPE(CMT_WITHDRAWAL, CMS_REQUEST), cnp::MT_WITHDRAWAL_RESPONSE = MAKE_MSG_TYPE(CMT_WITHDRAWAL, CMS_RESPONSE),

cnp::MT_BALANCE_QUERY_REQUEST = MAKE_MSG_TYPE(CMT_BALANCE_QUERY, CMS_REQUEST), cnp::MT_BALANCE_QUERY_RESPONSE = MAKE_MSG_TYPE(CMT_BALANCE_QUERY, CMS_RESPONSE), cnp::MT_TRANSACTION_QUERY_REQUEST = MAKE_MSG_TYPE(CMT_TRANSACTION_QUERY, CMS_REQUEST), cnp::MT_TRANSACTION_QUERY_RESPONSE = MAKE_MSG_TYPE(CMT_TRANSACTION_QUERY, CMS_RESPONSE), cnp::MT_PURCHASE_STAMPS_REQUEST = MAKE_MSG_TYPE(CMT_PURCHASE_STAMPS, CMS_REQUEST), cnp::MT_PURCHASE_STAMPS_RESPONSE = MAKE_MSG_TYPE(CMT_PURCHASE_STAMPS, CMS_RESPONSE) }

- *Constructed Message Type IDs.* enum cnp::CFC_TYPE { cnp::CFC_CONNECT, cnp::CFC_CREDENTIALS, cnp::CFC_FUNCTIONAL, cnp::CFC_ACCOUNT, cnp::CFC_UNDEFINED }
- *Message Facility Code Types (CFC)* enum cnp::DEPOSIT_TYPE { cnp::DT_INVALID = 0, cnp::DT_CASH = 0x01, cnp::DT_CHECK = 0x02 }
- *CNP Deposit types (DT)* enum cnp::TRANSACTION_TYPE { cnp::TT_INVALID = 0, cnp::TT_DEPOSIT = 0x01, cnp::TT_WITHDRAWAL = 0x02, cnp::TT_STAMP_PURCHASE = 0x03 }

CNP Transaction types (TT)

Detailed Description

Enumeration Type Documentation

enum cnp::MSG_TYPE

Enumerator

MT_INVALID
MT_CONNECT_REQUEST
MT_CONNECT_RESPONSE
MT_CREATE_ACCOUNT_REQUEST
MT_CREATE_ACCOUNT_RESPONSE
MT_LOGON_REQUEST
MT_LOGON_RESPONSE
MT_LOGOFF_REQUEST
MT_LOGOFF_RESPONSE
MT_DEPOSIT_REQUEST
MT_DEPOSIT_RESPONSE
MT_WITHDRAWAL_REQUEST
MT_WITHDRAWAL_RESPONSE
MT_BALANCE_QUERY_REQUEST
MT_BALANCE_QUERY_RESPONSE
MT_TRANSACTION_QUERY_REQUEST
MT_TRANSACTION_QUERY_RESPONSE
MT_PURCHASE_STAMPS_REQUEST
MT_PURCHASE_STAMPS_RESPONSE

enum cnp::CFC_TYPE

Used in the creation of result codes returned to the client. The purpose is to help provide the client useful diagnostic information regarding associating specific errors with a particular facility or functional subsystem.

Enumerator

CFC_CONNECT Connection validation related issues.

CFC_CREDENTIALS Logon related issues.

CFC_FUNCTIONAL Invalid arguments or Client state.

CFC_ACCOUNT Account related errors related to balances, etc.

CFC_UNDEFINED Other error categories not explicitly defined.

enum cnp::DEPOSIT_TYPE

Enumerator

DT_INVALID for initialization and error checking

DT_CASH Cash Deposit.

DT_CHECK Check Deposit.

enum cnp::TRANSACTION_TYPE

Enumerator

TT_INVALID for initialization and error checking

TT_DEPOSIT Deposit Transaction.

TT_WITHDRAWAL Withdrawal Transaction.

TT_STAMP_PURCHASE Stamp Purchase Transaction.

Namespace Documentation

cnp Namespace Reference

Namespaces

- prim

Classes

- struct BALANCE_QUERY_REQUEST
- *[Client] Balance Query Request message* struct BALANCE_QUERY_RESPONSE
- *[Server] Balance Query Response message* struct CONNECT_REQUEST
- *[Client] Connect Request message* struct CONNECT_RESPONSE
- *[Server] Connect Response message* struct CREATE_ACCOUNT_REQUEST
- *[Client] Create Account Request message* struct CREATE_ACCOUNT_RESPONSE
- *[Server] Create Account Response message* struct DEPOSIT_REQUEST
- *[Client] Deposit Request message* struct DEPOSIT_RESPONSE
- *[Server] Deposit Response message* struct LOGOFF_REQUEST
- *[Client] Logoff Request message* struct LOGOFF_RESPONSE
- *[Server] Logoff Response message* struct LOGON_REQUEST
- *[Client] Logon Request message* struct LOGON_RESPONSE
- *[Server] Logon Response message* struct STAMP_PURCHASE_REQUEST
- *[Client] Stamp Purchase Request Message* struct STAMP_PURCHASE_RESPONSE

- *[Server] Stamp Purchase Response message* struct STD_HDR
- *CNP Standard Message Header.* struct TRANSACTION
- *A Customer Transaction Record.* struct TRANSACTION_QUERY_REQUEST
- *[Client] Transaction Query Request message* struct TRANSACTION_QUERY_RESPONSE
- *[Server] Transaction Query Response message* struct WITHDRAWAL_REQUEST
- *[Client] Withdrawal Request message* struct WITHDRAWAL_RESPONSE

[Server] Withdrawal Response message Typedefs

- typedef unsigned short WORD
  16bit type definition
- typedef unsigned long DWORD
  32bit type definition
- typedef unsigned long long QWORD
  64bit type definition

Enumerations

- enum CNP_MSG_TYPE { CMT_INVALID = 0x00, CMT_CONNECT = 0x50, CMT_CREATE_ACCOUNT = 0x51, CMT_LOGON = 0x52, CMT_LOGOFF = 0x53, CMT_DEPOSIT = 0x54, CMT_WITHDRAWAL = 0x55, CMT_BALANCE_QUERY = 0x56, CMT_TRANSACTION_QUERY = 0x57, CMT_PURCHASE_STAMPS = 0x58 }
- *Supported CNP Message Types (CMT_)* enum CNP_MSG_SUBTYPE { CMS_INVALID = 0x00, CMS_REQUEST = 0x01, CMS_RESPONSE = 0x02 }
- *Supported CNP Message Subtypes (CMS_)* enum MSG_TYPE { MT_INVALID = MAKE_MSG_TYPE(CMT_INVALID, CMS_INVALID), MT_CONNECT_REQUEST = MAKE_MSG_TYPE(CMT_CONNECT, CMS_REQUEST), MT_CONNECT_RESPONSE = MAKE_MSG_TYPE(CMT_CONNECT, CMS_RESPONSE), MT_CREATE_ACCOUNT_REQUEST = MAKE_MSG_TYPE(CMT_CREATE_ACCOUNT, CMS_REQUEST), MT_CREATE_ACCOUNT_RESPONSE = MAKE_MSG_TYPE(CMT_CREATE_ACCOUNT, CMS_RESPONSE), MT_LOGON_REQUEST = MAKE_MSG_TYPE(CMT_LOGON, CMS_REQUEST), MT_LOGON_RESPONSE = MAKE_MSG_TYPE(CMT_LOGON, CMS_RESPONSE), MT_LOGOFF_REQUEST = MAKE_MSG_TYPE(CMT_LOGOFF, CMS_REQUEST), MT_LOGOFF_RESPONSE = MAKE_MSG_TYPE(CMT_LOGOFF, CMS_RESPONSE), MT_DEPOSIT_REQUEST = MAKE_MSG_TYPE(CMT_DEPOSIT, CMS_REQUEST), MT_DEPOSIT_RESPONSE = MAKE_MSG_TYPE(CMT_DEPOSIT, CMS_RESPONSE), MT_WITHDRAWAL_REQUEST = MAKE_MSG_TYPE(CMT_WITHDRAWAL, CMS_REQUEST), MT_WITHDRAWAL_RESPONSE = MAKE_MSG_TYPE(CMT_WITHDRAWAL, CMS_RESPONSE), MT_BALANCE_QUERY_REQUEST = MAKE_MSG_TYPE(CMT_BALANCE_QUERY, CMS_REQUEST), MT_BALANCE_QUERY_RESPONSE = MAKE_MSG_TYPE(CMT_BALANCE_QUERY, CMS_RESPONSE), MT_TRANSACTION_QUERY_REQUEST = MAKE_MSG_TYPE(CMT_TRANSACTION_QUERY, CMS_REQUEST), MT_TRANSACTION_QUERY_RESPONSE = MAKE_MSG_TYPE(CMT_TRANSACTION_QUERY, CMS_RESPONSE), MT_PURCHASE_STAMPS_REQUEST = MAKE_MSG_TYPE(CMT_PURCHASE_STAMPS, CMS_REQUEST), MT_PURCHASE_STAMPS_RESPONSE = MAKE_MSG_TYPE(CMT_PURCHASE_STAMPS, CMS_RESPONSE) }
- *Constructed Message Type IDs.* enum CFC_TYPE { CFC_CONNECT, CFC_CREDENTIALS, CFC_FUNCTIONAL, CFC_ACCOUNT, CFC_UNDEFINED }
- *Message Facility Code Types (CFC)* enum CER_TYPE { CER_SUCCESS = 0, CER_AUTHENICATION_FAILED = MAKE_ERROR_RESULT(CFC_CONNECT, 0x01), CER_UNSUPPORTED_PROTOCOL = MAKE_ERROR_RESULT(CFC_CONNECT, 0x02), CER_INVALID_CLIENT_ID = MAKE_ERROR_RESULT(CFC_CREDENTIALS, 0x01), CER_INVALID_NAME_PIN = MAKE_ERROR_RESULT(CFC_CREDENTIALS, 0x02), CER_INVALID_ARGUMENTS = MAKE_ERROR_RESULT(CFC_FUNCTIONAL, 0x01),

CER_CLIENT_NOT_LOGGEDON = MAKE_ERROR_RESULT(CFC_FUNCTIONAL, 0x02),
CER_DRAWER_BLOCKED = MAKE_ERROR_RESULT(CFC_FUNCTIONAL, 0x03),
CER_INSUFFICIENT_FUNDS = MAKE_ERROR_RESULT(CFC_ACCOUNT, 0x01),
CER_ACCOUNT_NOT_FOUND = MAKE_ERROR_RESULT(CFC_ACCOUNT, 0x02),
CER_ACCOUNT_EXISTS = MAKE_ERROR_RESULT(CFC_ACCOUNT, 0x03), CER_ERROR = (~0) }

- *CNP Error Result Types (CER)* enum DEPOSIT_TYPE { DT_INVALID = 0, DT_CASH = 0x01, DT_CHECK = 0x02 }
- *CNP Deposit types (DT)* enum TRANSACTION_TYPE { TT_INVALID = 0, TT_DEPOSIT = 0x01, TT_WITHDRAWAL = 0x02, TT_STAMP_PURCHASE = 0x03 }

CNP Transaction types (TT) Functions

- bool Succeeded (cnp::CER_TYPE cerRR) throw ()

Variables

- const WORD g_wMajorVersion = 1
  Global message sequence number.
- const WORD g_wMinorVersion = 1
  Protocol minor version (i.e. x.3)
- const DWORD g_dwValidationKey = 0x00DEAD01
  CNP Validation Key.
- const size_t MAX_NAME_LEN = 32
  [first,last,email] name field lengths
- const WORD INVALID_CLIENT_ID = (~0)
  Used for error checking & default initialization.
- const WORD INVALID_PIN = 0
  Used for error checking & default initialization.

Typedef Documentation

typedef unsigned short cnp::WORD

typedef unsigned long cnp::DWORD

typedef unsigned long long cnp::QWORD

Enumeration Type Documentation

enum cnp::CNP_MSG_TYPE

Enumerator

CMT_INVALID used for initialization and error checking

CMT_CONNECT
CMT_CREATE_ACCOUNT
CMT_LOGON
CMT_LOGOFF
CMT_DEPOSIT

CMT_WITHDRAWAL
CMT_BALANCE_QUERY
CMT_TRANSACTION_QUERY
CMT_PURCHASE_STAMPS

enum cnp::CNP_MSG_SUBTYPE

Enumerator

CMS_INVALID used for initialization and error checking

CMS_REQUEST
CMS_RESPONSE

enum cnp::CER_TYPE

Enumerator

CER_SUCCESS Success!

CER_AUTHENICATION_FAILED Invalid validation key.

CER_UNSUPPORTED_PROTOCOL Protocol version not supported.

CER_INVALID_CLIENT_ID Invalid client ID found.

CER_INVALID_NAME_PIN Invalid name or pin.

CER_INVALID_ARGUMENTS Invalid arguments used.

CER_CLIENT_NOT_LOGGEDON Client not logged-on.

CER_DRAWER_BLOCKED Mechanical Failure.

CER_INSUFFICIENT_FUNDS Insufficient funds available.

CER_ACCOUNT_NOT_FOUND Client account does not exist.

CER_ACCOUNT_EXISTS Prior account already exists.

CER_ERROR Generic error result.

Function Documentation

bool cnp::Succeeded (cnp::CER_TYPE *cerRR*) throw)

Variable Documentation

const WORD cnp::g_wMajorVersion = 1

The following is a little complicated, but I will try to explain what is going on.

g_dwSequenceNumber is a static global variable intended to be auto-incremented by the client as part of constructing request messages. It needs to be defined such that it is accessible by various message constructors, but at the same time does not create a linker error as it can be included in multiple .cpp files. (resulting in multiple instances across various COMDATs)

So, we are telling the linker to just use one of the instances here if it is finds multiple instance declarations.

Another complication is that MSVC++ & GNUC++ cannot agree on how to do this.

See also:

https://gcc.gnu.org/onlinedocs/gcc/Vague-Linkage.html
https://msdn.microsoft.com/EN-US/library/5tkz6s71(v=VS.120,d=hv.2).aspxCNP Protocol version
Protocol major version (i.e. 1.x)

const WORD cnp::g_wMinorVersion = 1

const DWORD cnp::g_dwValidationKey = 0x00DEAD01

const size_t cnp::MAX_NAME_LEN = 32

See also:

CNP_CREATE_ACCOUNT_REQUEST, CNP_LOGON_REQUEST

const WORD cnp::INVALID_CLIENT_ID = (~0)

const WORD cnp::INVALID_PIN = 0

---

cnp::prim Namespace Reference

Classes

- struct _BALANCE_QUERY_REQUEST
- *Balance Query Request Primitive.* struct _BALANCE_QUERY_RESPONSE
- *Balance Query Response Primitive.* struct _CONNECT_REQUEST
- *Connect Request Primitive.* struct _CONNECT_RESPONSE
- *Connection Response Primitive.* struct _CREATE_ACCOUNT_REQUEST
- *Create Account Request Primitive.* struct _CREATE_ACCOUNT_RESPONSE
- *Create Account Response Primitive.* struct _DEPOSIT_REQUEST
- *Deposit Request Primitive.* struct _DEPOSIT_RESPONSE
- *Deposit Response Primitive.* struct _LOGOFF_REQUEST
- *Logoff Request Primitive.* struct _LOGOFF_RESPONSE
- *Logoff Response Primitive.* struct _LOGON_REQUEST
- *Logon Request Primitive.* struct _LOGON_RESPONSE
- *Logon Response Primitive.* struct _STAMP_PURCHASE_REQUEST
- *Purchase Stamp Request Primitive.* struct _STAMP_PURCHASE_RESPONSE
- *Stamp Purchase Response Primitive.* struct _TRANSACTION_QUERY_REQUEST
- *Transaction Query Request Primitive.* struct _TRANSACTION_QUERY_RESPONSE
- *Transaction Query Result Primitive.* struct _WITHDRAWAL_REQUEST
- *Withdrawal Request Primitive.* struct _WITHDRAWAL_RESPONSE

Withdrawal Response Primitive.

Class Documentation

cnp::prim::_BALANCE_QUERY_REQUEST Struct Reference

Balance Query Request Primitive.

Public Member Functions

- _BALANCE_QUERY_REQUEST ()

Constructor & Destructor Documentation

cnp::prim::_BALANCE_QUERY_REQUEST::_BALANCE_QUERY_REQUEST ()

---

cnp::prim::_BALANCE_QUERY_RESPONSE Struct Reference

Balance Query Response Primitive.

Public Member Functions

- _BALANCE_QUERY_RESPONSE (DWORD dwResult=cnp::CER_ERROR, DWORD dwBalance=0)
- DWORD get_Balance (void) const

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.
- DWORD m_dwBalance
  Current Client account balance.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_BALANCE_QUERY_RESPONSE::_BALANCE_QUERY_RESPONSE (DWORD *dwResult* = `cnp::CER_ERROR`, DWORD *dwBalance* = 0)

Member Function Documentation

DWORD cnp::prim::_BALANCE_QUERY_RESPONSE::get_Balance (void) const

---

Member Data Documentation

DWORD cnp::prim::_BALANCE_QUERY_RESPONSE::m_dwResult

DWORD cnp::prim::_BALANCE_QUERY_RESPONSE::m_dwBalance

---

cnp::prim::_CONNECT_REQUEST Struct Reference

Connect Request Primitive.

Public Member Functions

- _CONNECT_REQUEST ()
  Default constructor.
- _CONNECT_REQUEST (WORD wMajorVersion, WORD wMinorVersion, DWORD dwKey)
  Initialization constructor.

Public Attributes

- WORD m_wMajorVersion
  Client Major Protocol version number.
- WORD m_wMinorVersion
  Client Minor Protocol version number.
- DWORD m_dwValidationKey
  Used by Server to authenticate the connection.

Detailed Description

Field(s)	Begin Byte	End Byte
m_wMajorVersion	0	1
m_wMinorVersion	2	3
m_dwValidationKey	4	7

Constructor & Destructor Documentation

cnp::prim::_CONNECT_REQUEST::_CONNECT_REQUEST ()

cnp::prim::_CONNECT_REQUEST::_CONNECT_REQUEST (WORD *wMajorVersion*, WORD *wMinorVersion*, DWORD *dwKey*)

Member Data Documentation

WORD cnp::prim::_CONNECT_REQUEST::m_wMajorVersion

WORD cnp::prim::_CONNECT_REQUEST::m_wMinorVersion

DWORD cnp::prim::_CONNECT_REQUEST::m_dwValidationKey

---

cnp::prim::_CONNECT_RESPONSE Struct Reference

Connection Response Primitive.

Public Member Functions

- _CONNECT_RESPONSE (DWORD dwResult=cnp::CER_ERROR, WORD wMajorVersion=0, WORD wMinorVersion=0, WORD wClientID=INVALID_CLIENT_ID)
  Initialization Constructor.

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.
- WORD m_wMajorVersion
  Server Major Protocol version number.
- WORD m_wMinorVersion
  Server Minor Protocol version number.
- WORD m_wClientID

Detailed Description

Field(s)	Begin Byte	End Byte
m_dwResult	0	3
m_wMajorVersion	4	5
m_wMinorVersion	6	7
m_wClientID	8	9

See also:

cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_CONNECT_RESPONSE::_CONNECT_RESPONSE (DWORD *dwResult* = cnp::CER_ERROR, WORD *wMajorVersion* = 0, WORD *wMinorVersion* = 0, WORD *wClientID* = INVALID_CLIENT_ID)

Parameters:

in	*dwResult*	Server generated cnp::CER_TYPE result
in	*wMajorVersion*	Current Server major protocol version number
in	*wMinorVersion*	Current Server minor protocol version number
in	*wClientID*	Server generated ClientID

Member Data Documentation

DWORD cnp::prim::_CONNECT_RESPONSE::m_dwResult

WORD cnp::prim::_CONNECT_RESPONSE::m_wMajorVersion

WORD cnp::prim::_CONNECT_RESPONSE::m_wMinorVersion

WORD cnp::prim::_CONNECT_RESPONSE::m_wClientID

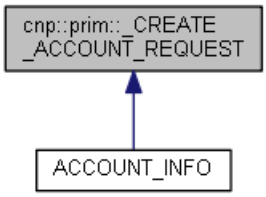
generated by the Server and is required in all subsequent request messages by the Client

---

cnp::prim::_CREATE_ACCOUNT_REQUEST Struct Reference

Create Account Request Primitive.

Inheritance diagram for cnp::prim::_CREATE_ACCOUNT_REQUEST:



Public Member Functions

- _CREATE_ACCOUNT_REQUEST ()
  Default constructor.

- _CREATE_ACCOUNT_REQUEST (const char *szFirstName, const char *szLastName, const char *szEmailAddress, WORD wPIN, DWORD dwSSNumber, DWORD dwDLNumber)
  Initialization constructor.
- void set_FirstName (const char *szSet)
- void set_LastName (const char *szSet)
- void set_EmailAddress (const char *szSet)

Public Attributes

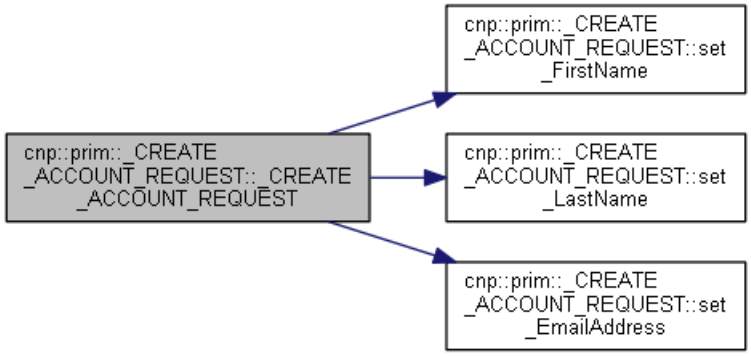
- char m_szFirstName [MAX_NAME_LEN]
  User's First Name.
- char m_szLastName [MAX_NAME_LEN]
  User's Last Name.
- char m_szEmailAddress [MAX_NAME_LEN]
  User's Email Address.
- WORD m_wPIN
  User's Personal Identification Number.
- DWORD m_dwSSNumber
  (optional) User's Social Security Number
- DWORD m_dwDLNumber
  (optional) User's Drivers License Number

Constructor & Destructor Documentation

cnp::prim::_CREATE_ACCOUNT_REQUEST::_CREATE_ACCOUNT_REQUEST ()

cnp::prim::_CREATE_ACCOUNT_REQUEST::_CREATE_ACCOUNT_REQUEST (const char * *szFirstName*, const char * *szLastName*, const char * *szEmailAddress*, WORD *wPIN*, DWORD *dwSSNumber*, DWORD *dwDLNumber*)

Here is the call graph for this function:



Member Function Documentation

void cnp::prim::_CREATE_ACCOUNT_REQUEST::set_FirstName (const char * *szSet*)

Parameters:

in	*szSet*	address containing null-terminated first name

void cnp::prim::_CREATE_ACCOUNT_REQUEST::set_LastName (const char * *szSet*)

Parameters:

in	*szSet*	address containing null-terminated last name

void cnp::prim::_CREATE_ACCOUNT_REQUEST::set_EmailAddress (const char * *szSet*)

Parameters:

in	*szSet*	address containing null-terminated email address

Member Data Documentation

char cnp::prim::_CREATE_ACCOUNT_REQUEST::m_szFirstName[MAX_NAME_LEN]

char cnp::prim::_CREATE_ACCOUNT_REQUEST::m_szLastName[MAX_NAME_LEN]

char cnp::prim::_CREATE_ACCOUNT_REQUEST::m_szEmailAddress[MAX_NAME_LEN]

WORD cnp::prim::_CREATE_ACCOUNT_REQUEST::m_wPIN

DWORD cnp::prim::_CREATE_ACCOUNT_REQUEST::m_dwSSNumber

DWORD cnp::prim::_CREATE_ACCOUNT_REQUEST::m_dwDLNumber

---

cnp::prim::_CREATE_ACCOUNT_RESPONSE Struct Reference

Create Account Response Primitive.

Public Member Functions

- _CREATE_ACCOUNT_RESPONSE (DWORD dwResult=cnp::CER_ERROR)

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_CREATE_ACCOUNT_RESPONSE::_CREATE_ACCOUNT_RESPONSE (DWORD *dwResult* = `cnp::CER_ERROR`)

Member Data Documentation

DWORD cnp::prim::_CREATE_ACCOUNT_RESPONSE::m_dwResult

cnp::prim::_DEPOSIT_REQUEST Struct Reference

Deposit Request Primitive.

Public Member Functions

- _DEPOSIT_REQUEST (DWORD dwAmount=0, DEPOSIT_TYPE Type=DT_INVALID)
  Initialization constructor.

Public Attributes

- DWORD m_dwAmount
  Amount excluding decimal point (i.e. $100.00 would be 10000)
- WORD m_wType
  cnp::DT_CASH or cnp::DT_CHECK

Detailed Description

See also:
cnp::DEPOSIT_TYPE

Constructor & Destructor Documentation

cnp::prim::_DEPOSIT_REQUEST::_DEPOSIT_REQUEST (DWORD *dwAmount* = 0, DEPOSIT_TYPE *Type* = DT_INVALID)

Member Data Documentation

DWORD cnp::prim::_DEPOSIT_REQUEST::m_dwAmount

WORD cnp::prim::_DEPOSIT_REQUEST::m_wType

---

cnp::prim::_DEPOSIT_RESPONSE Struct Reference

Deposit Response Primitive.

Public Member Functions

- _DEPOSIT_RESPONSE (DWORD dwResult=cnp::CER_ERROR)

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_DEPOSIT_RESPONSE::_DEPOSIT_RESPONSE (DWORD *dwResult* = cnp::CER_ERROR)

Member Data Documentation

DWORD cnp::prim::_DEPOSIT_RESPONSE::m_dwResult

---

cnp::prim::_LOGOFF_REQUEST Struct Reference

Logoff Request Primitive.

Public Member Functions

- _LOGOFF_REQUEST ()

Constructor & Destructor Documentation

cnp::prim::_LOGOFF_REQUEST::_LOGOFF_REQUEST ()

---

cnp::prim::_LOGOFF_RESPONSE Struct Reference

Logoff Response Primitive.

Public Member Functions

- _LOGOFF_RESPONSE (DWORD dwResult=cnp::CER_ERROR)

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_LOGOFF_RESPONSE::_LOGOFF_RESPONSE (DWORD *dwResult* = cnp::CER_ERROR)

Member Data Documentation

DWORD cnp::prim::_LOGOFF_RESPONSE::m_dwResult

---

cnp::prim::_LOGON_REQUEST Struct Reference

Logon Request Primitive.

Public Member Functions

- _LOGON_REQUEST ()
  Default constructor.
- _LOGON_REQUEST (const char *szFirstName, WORD wPIN)
  Initialization constructor.
- void set_FirstName (const char *szSet)

Public Attributes

- char m_szFirstName [MAX_NAME_LEN]
  User's first name.
- WORD m_wPIN
  Personal Identification Number.

Constructor & Destructor Documentation

cnp::prim::_LOGON_REQUEST::_LOGON_REQUEST ()

cnp::prim::_LOGON_REQUEST::_LOGON_REQUEST (const char * *szFirstName*, WORD *wPIN*)

Here is the call graph for this function:

Member Function Documentation

void cnp::prim::_LOGON_REQUEST::set_FirstName (const char * *szSet*)

Parameters:

in	*szSet*	address containing null terminated first name

Member Data Documentation

char cnp::prim::_LOGON_REQUEST::m_szFirstName[MAX_NAME_LEN]

WORD cnp::prim::_LOGON_REQUEST::m_wPIN

cnp::prim::_LOGON_RESPONSE Struct Reference

Logon Response Primitive.

Public Member Functions

- _LOGON_RESPONSE (DWORD dwResult=cnp::CER_ERROR)

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_LOGON_RESPONSE::_LOGON_RESPONSE (DWORD *dwResult* = `cnp::CER_ERROR`)

Member Data Documentation

DWORD cnp::prim::_LOGON_RESPONSE::m_dwResult

cnp::prim::_STAMP_PURCHASE_REQUEST Struct Reference

Purchase Stamp Request Primitive.

Public Member Functions

- _STAMP_PURCHASE_REQUEST ()
  Default constructor.
- _STAMP_PURCHASE_REQUEST (DWORD dwAmount)
  Initialization constructor.

Public Attributes

- DWORD m_dwAmount
  Amount excluding decimal point (i.e. $100.00 would be 10000)

Constructor & Destructor Documentation

cnp::prim::_STAMP_PURCHASE_REQUEST::_STAMP_PURCHASE_REQUEST ()

cnp::prim::_STAMP_PURCHASE_REQUEST::_STAMP_PURCHASE_REQUEST (DWORD *dwAmount*)

Member Data Documentation

DWORD cnp::prim::_STAMP_PURCHASE_REQUEST::m_dwAmount

---

cnp::prim::_STAMP_PURCHASE_RESPONSE Struct Reference

Stamp Purchase Response Primitive.

Public Member Functions

- _STAMP_PURCHASE_RESPONSE (DWORD dwResult=cnp::CER_ERROR)

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

cnp::prim::_STAMP_PURCHASE_RESPONSE::_STAMP_PURCHASE_RESPONSE (DWORD *dwResult* = `cnp::CER_ERROR`)

Member Data Documentation

DWORD cnp::prim::_STAMP_PURCHASE_RESPONSE::m_dwResult

cnp::prim::_TRANSACTION_QUERY_REQUEST Struct Reference

Transaction Query Request Primitive.

Public Member Functions

- _TRANSACTION_QUERY_REQUEST ()
  Default constructor.
- _TRANSACTION_QUERY_REQUEST (DWORD dwStartID, WORD wTransactionCount)
  Initialization constructor.

Public Attributes

- DWORD m_dwStartID
  the transaction number to begin the query from
- WORD m_wTransactionCount
  the number of transactions requested

Constructor & Destructor Documentation

cnp::prim::_TRANSACTION_QUERY_REQUEST::_TRANSACTION_QUERY_REQUEST ()

cnp::prim::_TRANSACTION_QUERY_REQUEST::_TRANSACTION_QUERY_REQUEST (DWORD *dwStartID*, WORD *wTransactionCount*)

Member Data Documentation

DWORD cnp::prim::_TRANSACTION_QUERY_REQUEST::m_dwStartID

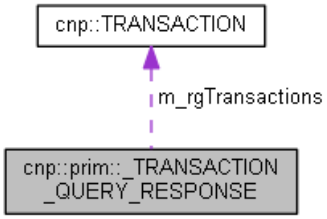
WORD cnp::prim::_TRANSACTION_QUERY_REQUEST::m_wTransactionCount

---

cnp::prim::_TRANSACTION_QUERY_RESPONSE Struct Reference

Transaction Query Result Primitive.

Collaboration diagram for cnp::prim::_TRANSACTION_QUERY_RESPONSE:



Public Member Functions

- _TRANSACTION_QUERY_RESPONSE (DWORD dwResult=cnp::CER_ERROR, WORD wTransactionCount=0)
- WORD get_TransactionCount (void) const

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.
- WORD m_wTransactionCount
  number of transactions returned in array
- TRANSACTION m_rgTransactions []
  unsized array of Transaction records

Detailed Description

See also:

cnp::CER_TYPE
cnp::TRANSACTION

Constructor & Destructor Documentation

cnp::prim::_TRANSACTION_QUERY_RESPONSE::_TRANSACTION_QUERY_RESPONSE (DWORD *dwResult* = `cnp::CER_ERROR`, WORD *wTransactionCount* = 0)

Member Function Documentation

WORD cnp::prim::_TRANSACTION_QUERY_RESPONSE::get_TransactionCount (void) const

---

Member Data Documentation

DWORD cnp::prim::_TRANSACTION_QUERY_RESPONSE::m_dwResult

WORD cnp::prim::_TRANSACTION_QUERY_RESPONSE::m_wTransactionCount

TRANSACTION cnp::prim::_TRANSACTION_QUERY_RESPONSE::m_rgTransactions[]

---

cnp::prim::_WITHDRAWAL_REQUEST Struct Reference

Withdrawal Request Primitive.

Public Member Functions

- _WITHDRAWAL_REQUEST ()
  Default constructor.
- _WITHDRAWAL_REQUEST (DWORD dwAmount)
  Initialization constructor.

Public Attributes

- DWORD m_dwAmount
  Amount excluding decimal point (i.e. $100.00 would be 10000)

Constructor & Destructor Documentation

cnp::prim::_WITHDRAWAL_REQUEST::_WITHDRAWAL_REQUEST ()

cnp::prim::_WITHDRAWAL_REQUEST::_WITHDRAWAL_REQUEST (DWORD *dwAmount*)

Member Data Documentation

DWORD cnp::prim::_WITHDRAWAL_REQUEST::m_dwAmount

---

cnp::prim::_WITHDRAWAL_RESPONSE Struct Reference

Withdrawal Response Primitive.

Public Member Functions

- _WITHDRAWAL_RESPONSE (DWORD dwResult=cnp::CER_ERROR)

Public Attributes

- DWORD m_dwResult
  Success or Error code from cnp::CER_TYPE.

Detailed Description

See also:
cnp::CER_TYPE

Constructor & Destructor Documentation

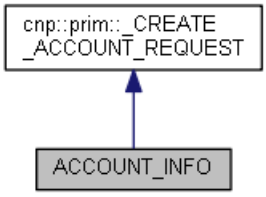
cnp::prim::_WITHDRAWAL_RESPONSE::_WITHDRAWAL_RESPONSE (DWORD *dwResult* = cnp::CER_ERROR)

Member Data Documentation

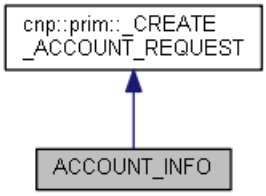
DWORD cnp::prim::_WITHDRAWAL_RESPONSE::m_dwResult

ACCOUNT_INFO Struct Reference

Inheritance diagram for ACCOUNT_INFO:



Collaboration diagram for ACCOUNT_INFO:



Public Types

- typedef cnp::QWORD key_type
- typedef cnp::prim::_CREATE_ACCOUNT_REQUEST _Base

Public Member Functions

- ACCOUNT_INFO ()
  Default Constructor.
- ACCOUNT_INFO (const _Base &base, const cnp::QWORD &qwID=INVALID_CUSTOMER_ID, cnp::DWORD dwBalance=INVALID_BALANCE)
  Initialization Constructor.
- ACCOUNT_INFO (const ACCOUNT_INFO &rhs)
  Copy Constructor.
- const cnp::QWORD & get_PrimaryKey (void) const throw ()
- const cnp::QWORD & get_CustomerID (void) const throw ()
- cnp::DWORD get_Balance (void) const throw ()

- void set_Balance (cnp::DWORD dwSet) throw ()
- void decr_Balance (cnp::DWORD dwSet) throw ()
- void incr_Balance (cnp::DWORD dwSet) throw ()

Public Attributes

- cnp::QWORD m_qwCustomerID
- cnp::DWORD m_dwBalance

Detailed Description

ACCOUNT_INFO is used to maintain and persist information as it relates to an individual customer. It uses the Customer ID as the key field.

Member Typedef Documentation

typedef cnp::QWORD ACCOUNT_INFO::key_type

typedef cnp::prim::_CREATE_ACCOUNT_REQUEST ACCOUNT_INFO::_Base

---

Constructor & Destructor Documentation

ACCOUNT_INFO::ACCOUNT_INFO ()

ACCOUNT_INFO::ACCOUNT_INFO (const _Base & *base*, const cnp::QWORD & *qwID* = INVALID_CUSTOMER_ID, cnp::DWORD *dwBalance* = INVALID_BALANCE)

ACCOUNT_INFO::ACCOUNT_INFO (const ACCOUNT_INFO & *rhs*)

Member Function Documentation

const cnp::QWORD& ACCOUNT_INFO::get_PrimaryKey (void) const throw)

This method is used to provide a generic interface to retrieve a record's primary key field. In this instance, it is a thin wrapper around get_CustomerID().

Return values:

cnp::QWORD	containing the primary key value

Here is the call graph for this function:

const cnp::QWORD& ACCOUNT_INFO::get_CustomerID (void) const throw)

cnp::DWORD ACCOUNT_INFO::get_Balance (void) const throw)

void ACCOUNT_INFO::set_Balance (cnp::DWORD *dwSet*) throw)

void ACCOUNT_INFO::decr_Balance (cnp::DWORD *dwSet*) throw)

void ACCOUNT_INFO::incr_Balance (cnp::DWORD *dwSet*) throw)

Member Data Documentation

cnp::QWORD ACCOUNT_INFO::m_qwCustomerID

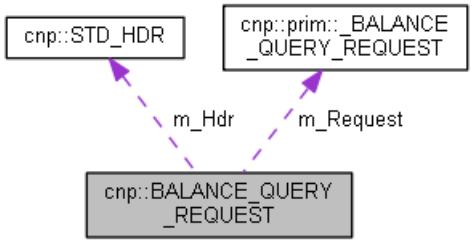
cnp::DWORD ACCOUNT_INFO::m_dwBalance

---

cnp::BALANCE_QUERY_REQUEST Struct Reference

[Client] Balance Query Request message

Collaboration diagram for cnp::BALANCE_QUERY_REQUEST:



Public Member Functions

- BALANCE_QUERY_REQUEST ()
  Default constructor.
- BALANCE_QUERY_REQUEST (WORD wClientID, DWORD dwContext=0)
  Initialization constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_BALANCE_QUERY_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15

Constructor & Destructor Documentation

cnp::BALANCE_QUERY_REQUEST::BALANCE_QUERY_REQUEST ()

cnp::BALANCE_QUERY_REQUEST::BALANCE_QUERY_REQUEST (WORD *wClientID*, DWORD *dwContext* = 0)

Note:

auto increments the Client's global sequence number

Parameters:

wClientID	Server generated Client ID
dwContext	[Optional] Client provided field

Member Function Documentation

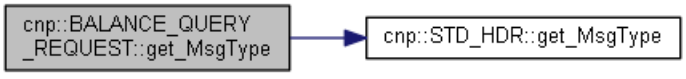
size_t cnp::BALANCE_QUERY_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

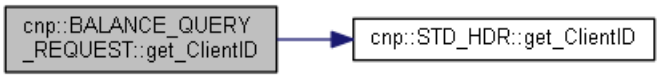
DWORD cnp::BALANCE_QUERY_REQUEST::get_MsgType (void) const

Here is the call graph for this function:



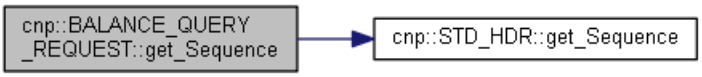
WORD cnp::BALANCE_QUERY_REQUEST::get_ClientID (void) const

Here is the call graph for this function:



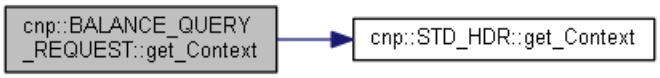
DWORD cnp::BALANCE_QUERY_REQUEST::get_Sequence (void) const

Here is the call graph for this function:



DWORD cnp::BALANCE_QUERY_REQUEST::get_Context (void) const

Here is the call graph for this function:



Member Data Documentation

STD_HDR cnp::BALANCE_QUERY_REQUEST::m_Hdr

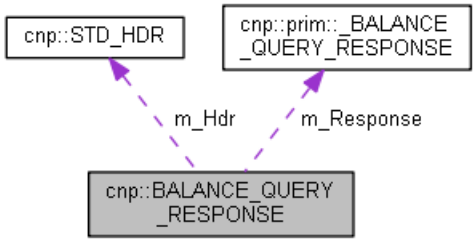
prim::_BALANCE_QUERY_REQUEST cnp::BALANCE_QUERY_REQUEST::m_Request

---

cnp::BALANCE_QUERY_RESPONSE Struct Reference

[Server] Balance Query Response message

Collaboration diagram for cnp::BALANCE_QUERY_RESPONSE:



Public Member Functions

- BALANCE_QUERY_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwBalance, DWORD dwContext, DWORD dwSequence)
  Initialization Constructor.
- DWORD get_MsgType (void) const

- DWORD get_ResponseResult (void) const
- DWORD get_Balance (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_BALANCE_QUERY_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19
m_Response	m_dwBalance	20	23

See also:

cnp::BALANCE_QUERY_REQUEST

Constructor & Destructor Documentation

cnp::BALANCE_QUERY_RESPONSE::BALANCE_QUERY_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwBalance*, DWORD *dwContext*, DWORD *dwSequence*)

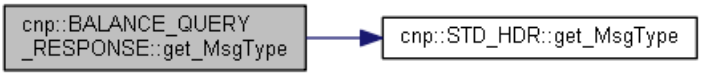
Parameters:

dwResult	Server generated cnp::CER_TYPE result
wClientID	Copied from BALANCE_QUERY_REQUEST
dwBalance	Client's current account balance
dwContext	Copied from BALANCE_QUERY_REQUEST
dwSequence	Copied from BALANCE_QUERY_REQUEST

Member Function Documentation

DWORD cnp::BALANCE_QUERY_RESPONSE::get_MsgType (void) const

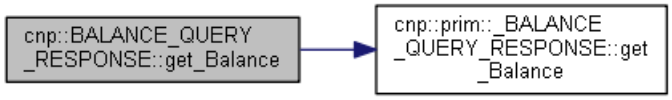
Here is the call graph for this function:



DWORD cnp::BALANCE_QUERY_RESPONSE::get_ResponseResult (void) const

DWORD cnp::BALANCE_QUERY_RESPONSE::get_Balance (void) const

Here is the call graph for this function:



size_t cnp::BALANCE_QUERY_RESPONSE::get_Size (void) const

Member Data Documentation

STD_HDR cnp::BALANCE_QUERY_RESPONSE::m_Hdr

prim::_BALANCE_QUERY_RESPONSE cnp::BALANCE_QUERY_RESPONSE::m_Response

---

CNP_Socket Class Reference

Public Member Functions

- CNP_Socket ()
  Default Constructor.
- CNP_Socket (SOCKET hSocket, const sockaddr_in &remoteAddr)
  Initialization Constructor.
- ~CNP_Socket ()
- void Close (void)
- bool Create (unsigned short wPort)
- bool Connect (const char *szHostAddress, unsigned short wPort)
- bool Listen (int iBackLog)
- bool Accept (SOCKET &hSocket, sockaddr_in &remoteAddr)
- int Receive (void *pData, size_t cbLen, int iFlags=0)
  Receives data from the underlying connected socket or a bound connectionless socket.
- int Send (const void *pData, size_t cbLen, int iFlags=0)
- int SetSocketOption (int iLevel, int iOption, const void *pVal, size_t cbLen)
  Sets the underlying socket option.
- bool SetBlocking (bool bBlocking=true)
- bool Shutdown (int iHow)
  disables sends or receives on underlying socket
- int GetError (void) const throw ()
- CNP_Socket ()
  Default Constructor.
- CNP_Socket (SOCKET hSocket, const sockaddr_in &remoteAddr)

Initialization Constructor.

- ~CNP_Socket ()
- void Close (void)
- bool Create (unsigned short wPort)
- bool Connect (const char *szHostAddress, unsigned short wPort)
- bool Listen (int iBackLog)
- bool Accept (SOCKET &hSocket, sockaddr_in &remoteAddr)
- int Receive (void *pData, size_t cbLen, int iFlags=0)
  Receives data from the underlying connected socket or a bound connectionless socket.
- int Send (const void *pData, size_t cbLen, int iFlags=0)
- int SetSocketOption (int iLevel, int iOption, const void *pVal, size_t cbLen)
  Sets the underlying socket option.
- bool SetBlocking (bool bBlocking=true)
- bool Shutdown (int iHow)
  disables sends or receives on underlying socket
- int GetError (void) const throw ()

Private Attributes

- SOCKET m_hSocket
- unsigned short m_wPort
- sockaddr_in m_LocalAddr
- sockaddr_in m_RemoteAddr
- int m_iError

Constructor & Destructor Documentation

CNP_Socket::CNP_Socket (void)

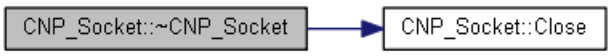
Performs member data initialization to default values only

CNP_Socket::CNP_Socket (SOCKET *hSocket*, const sockaddr_in & *remoteAddr*)

Performs member data initialization to default values, with the exception of the parameters

CNP_Socket::~CNP_Socket (void)

Here is the call graph for this function:



CNP_Socket::CNP_Socket ()

Performs member data initialization to default values only

CNP_Socket::CNP_Socket (SOCKET *hSocket*, const sockaddr_in & *remoteAddr*)

Performs member data initialization to default values, with the exception of the parameters

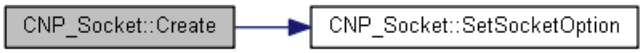
CNP_Socket::~CNP_Socket ()

Member Function Documentation

void CNP_Socket::Close (void)

bool CNP_Socket::Create (unsigned short *wPort*)

Here is the call graph for this function:



bool CNP_Socket::Connect (const char * *szHostAddress*, unsigned short *wPort*)

bool CNP_Socket::Listen (int *iBackLog*)

places the underlying socket in a state in which it is listening for an incoming connection

Parameters:

in	*iBackLog*	The maximum length of the queue of pending connections. If set to SOMAXCONN, the underlying service provider responsible for socket s will set the backlog to a maximum reasonable value. There is no standard provision to obtain the actual backlog value

Return values:

true	on success
false	on failure

bool CNP_Socket::Accept (SOCKET & *hSocket*, sockaddr_in & *remoteAddr*)

int CNP_Socket::Receive (void * *pData*, size_t *cbLen*, int *iFlags* = 0)

Parameters:

out	*pData*	A pointer to the buffer to receive the incoming data
in	*cbLen*	The length, in bytes, of the buffer pointed to by the pData parameter
in	*iFlags*	Optional parameter that influences the behavior of this function

Return values:

int	containing the number of bytes received and the buffer pointed to by the pData parameter will contain this data received
0	if the connection has been gracefully closed
SOCKET_ERROR	on failure call GetError() to retrieve the specific error code

int CNP_Socket::Send (const void * *pData*, size_t *cbLen*, int *iFlags* = 0)

int CNP_Socket::SetSocketOption (int *iLevel*, int *iOption*, const void * *pVal*, size_t *cbLen*)

Parameters:

in	*iLevel*	The level at which the option is defined (for example,

		SOL_SOCKET).
in	*iOption*	The socket option for which the value is to be set (for example, SO_BROADCAST). The iOption parameter must be a socket option defined within the specified level, or behavior is undefined
in	*pVal*	A pointer to the buffer in which the value for the requested option is specified
in	*cbLen*	The size, in bytes, of the buffer pointed to by the pVal parameter

Return values:

0	on success
SOCKET_ERROR	on failure call GetError() to retrieve the specific error code

bool CNP_Socket::SetBlocking (bool *bBlocking* = `true`)

bool CNP_Socket::Shutdown (int *iHow*)

Parameters:

in	*iHow*	A platform specific flag that describes what types of operation will no longer be allowed.

Return values:

true	on success
false	on failure

int CNP_Socket::GetError (void) const throw)

Return values:

int	containing the most recent error code

void CNP_Socket::Close (void)

bool CNP_Socket::Create (unsigned short *wPort*)

bool CNP_Socket::Connect (const char * *szHostAddress*, unsigned short *wPort*)

bool CNP_Socket::Listen (int *iBackLog*)

places the underlying socket in a state in which it is listening for an incoming connection

Parameters:

in	*iBackLog*	The maximum length of the queue of pending connections. If set to SOMAXCONN, the underlying service provider responsible for socket s will set the backlog to a maximum reasonable value. There is no standard provision to obtain the actual backlog value

Return values:

true	on success
false	on failure

bool CNP_Socket::Accept (SOCKET & *hSocket*, sockaddr_in & *remoteAddr*)

int CNP_Socket::Receive (void * *pData*, size_t *cbLen*, int *iFlags* = 0)

Parameters:

out	*pData*	A pointer to the buffer to receive the incoming data
in	*cbLen*	The length, in bytes, of the buffer pointed to by the pData parameter
in	*iFlags*	Optional parameter that influences the behavior of this function

Return values:

int	containing the number of bytes received and the buffer pointed to by the pData parameter will contain this data received
0	if the connection has been gracefully closed
SOCKET_ERROR	on failure call GetError() to retrieve the specific error code

int CNP_Socket::Send (const void * *pData*, size_t *cbLen*, int *iFlags* = 0)

int CNP_Socket::SetSocketOption (int *iLevel*, int *iOption*, const void * *pVal*, size_t *cbLen*)

Parameters:

in	*iLevel*	The level at which the option is defined (for example, SOL_SOCKET).
in	*iOption*	The socket option for which the value is to be set (for example, SO_BROADCAST). The iOption parameter must be a socket option defined within the specified level, or behavior is undefined
in	*pVal*	A pointer to the buffer in which the value for the requested option is specified
in	*cbLen*	The size, in bytes, of the buffer pointed to by the pVal parameter

Return values:

0	on success
SOCKET_ERROR	on failure call GetError() to retrieve the specific error code

bool CNP_Socket::SetBlocking (bool *bBlocking* = `true`)

bool CNP_Socket::Shutdown (int *iHow*)

Parameters:

in	*iHow*	A platform specific flag that describes what types of operation will no longer be allowed.

Return values:

true	on success
false	on failure

int CNP_Socket::GetError (void) const throw)

Return values:

int	containing the most recent error code

Member Data Documentation

SOCKET CNP_Socket::m_hSocket `[private]`

unsigned short CNP_Socket::m_wPort `[private]`

sockaddr_in CNP_Socket::m_LocalAddr `[private]`

sockaddr_in CNP_Socket::m_RemoteAddr `[private]`

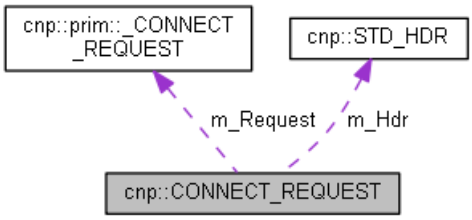
int CNP_Socket::m_iError `[private]`

---

cnp::CONNECT_REQUEST Struct Reference

[Client] Connect Request message

Collaboration diagram for cnp::CONNECT_REQUEST:



Public Member Functions

- CONNECT_REQUEST (WORD wClientID=0, WORD wMajorVersion=g_wMajorVersion, WORD wMinorVersion=g_wMinorVersion, DWORD dwValidationKey=g_dwValidationKey, DWORD dwContext=0) *Initialization Constructor.*
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- WORD get_ClientMajorVersion (void) const
- WORD get_ClientMinorVersion (void) const
- DWORD get_ClientValidationKey (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_CONNECT_REQUEST m_Request

Detailed Description

The connect request message establishes an authenticated connection with the Server. No Client ID is required for this message, but is provided by Server in CONNECT_RESPONSE message.

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_dwValidationKey	16	19
m_Request	m_wMajorVersion	20	21
m_Request	m_wMinorVersion	22	23

Constructor & Destructor Documentation

cnp::CONNECT_REQUEST::CONNECT_REQUEST (WORD *wClientID* = 0, WORD *wMajorVersion* = g_wMajorVersion, WORD *wMinorVersion* = g_wMinorVersion, DWORD *dwValidationKey* = g_dwValidationKey, DWORD *dwContext* = 0)

Parameters:

in	*wClientID*	[Optional] This value is ignored by server on initial connect
in	*wMajorVersion*	[Optional] Defaulted to g_wMajorVersion
in	*wMinorVersion*	[Optional] Defaulted to g_wMinorVersion
in	*dwValidationKey*	[Optional] Default to g_dwValidationKey
in	*dwContext*	[Optional] field for Client's use

Note:

auto increments the Client's global sequence number

Member Function Documentation

size_t cnp::CONNECT_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

DWORD cnp::CONNECT_REQUEST::get_MsgType (void) const

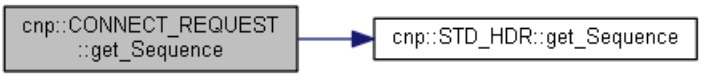
Here is the call graph for this function:

WORD cnp::CONNECT_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

DWORD cnp::CONNECT_REQUEST::get_Sequence (void) const

Here is the call graph for this function:



DWORD cnp::CONNECT_REQUEST::get_Context (void) const

Here is the call graph for this function:

WORD cnp::CONNECT_REQUEST::get_ClientMajorVersion (void) const

WORD cnp::CONNECT_REQUEST::get_ClientMinorVersion (void) const

DWORD cnp::CONNECT_REQUEST::get_ClientValidationKey (void) const

---

Member Data Documentation

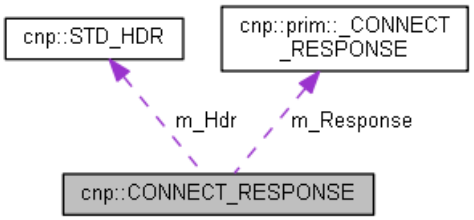
STD_HDR cnp::CONNECT_REQUEST::m_Hdr

prim::_CONNECT_REQUEST cnp::CONNECT_REQUEST::m_Request

---

cnp::CONNECT_RESPONSE Struct Reference

[Server] Connect Response message
Collaboration diagram for cnp::CONNECT_RESPONSE:



Public Member Functions

- CONNECT_RESPONSE ()
  Default Constructor.
- CONNECT_RESPONSE (DWORD dwResult, WORD wClientID, WORD wMajorVersion, WORD wMinorVersion, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_CONNECT_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19
m_Response	m_wMajorVersion	20	21
m_Response	m_wMinorVersion	22	23
m_Response	m_wClientID	24	25

See also:

cnp::CONNECT_REQUEST

Constructor & Destructor Documentation

cnp::CONNECT_RESPONSE::CONNECT_RESPONSE ()

cnp::CONNECT_RESPONSE::CONNECT_RESPONSE (DWORD *dwResult*, WORD *wClientID*, WORD *wMajorVersion*, WORD *wMinorVersion*, DWORD *dwSequence*, DWORD *dwContext*)

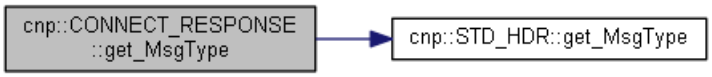
Parameters:

in	*dwResult*	Server generated cnp::CER_TYPE result
in	*wClientID*	Provided by Server on a successful connection response
in	*wMajorVersion*	current Server major version
in	*wMinorVersion*	current Server minor version
in	*dwSequence*	copied from CONNECT_REQUEST
in	*dwContext*	copied from CONNECT_REQUEST

Member Function Documentation

DWORD cnp::CONNECT_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:



WORD cnp::CONNECT_RESPONSE::get_ClientID (void) const

DWORD cnp::CONNECT_RESPONSE::get_ResponseResult (void) const

size_t cnp::CONNECT_RESPONSE::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

Member Data Documentation

STD_HDR cnp::CONNECT_RESPONSE::m_Hdr

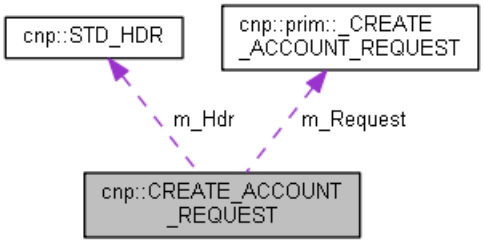
prim::_CONNECT_RESPONSE cnp::CONNECT_RESPONSE::m_Response

---

cnp::CREATE_ACCOUNT_REQUEST Struct Reference

[Client] Create Account Request message

Collaboration diagram for cnp::CREATE_ACCOUNT_REQUEST:



Public Member Functions

- CREATE_ACCOUNT_REQUEST ()
  Default constructor.
- CREATE_ACCOUNT_REQUEST (WORD wClientID, const char *szFirstName, const char *szLastName, const char *szEmailAddress, WORD wPIN, DWORD dwSSN=0, DWORD dwDLN=0, DWORD dwClientContext=0)
  Initialization constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- const char * get_FirstName (void) const
- const char * get_LastName (void) const
- const char * get_EmailAddress (void) const
- WORD get_PIN (void) const
- DWORD get_SSNumber (void) const
- DWORD get_DLNumber (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_CREATE_ACCOUNT_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_szFirsName	16	47
m_Request	m_szLastName	48	79
m_Request	m_szEmailAddress	80	111
m_Request	m_wPIN	112	113
m_Request	m_dwSSNumber	114	117
m_Request	m_dwDLNumber	118	121

Constructor & Destructor Documentation

cnp::CREATE_ACCOUNT_REQUEST::CREATE_ACCOUNT_REQUEST ()

cnp::CREATE_ACCOUNT_REQUEST::CREATE_ACCOUNT_REQUEST (WORD *wClientID*, const char * *szFirstName*, const char * *szLastName*, const char * *szEmailAddress*, WORD *wPIN*, DWORD *dwSSN* = 0, DWORD *dwDLN* = 0, DWORD *dwClientContext* = 0)

Parameters:

in	*wClientID*	Provided by Server on a successful connection response
in	*szFirstName*	Client's first name (used in LOGON messages)
in	*szLastName*	Client's last name
in	*szEmailAddress*	Client's email address
in	*wPIN*	Client Generated PIN (used in LOGON messages)
in	*dwSSN*	[Optional] Social Security Number field
in	*dwDLN*	[Optional] Driver's License Number field
in	*dwClientContext*	[Optional] field for the Client's use

Note:

auto increments the Client's global sequence number

Member Function Documentation

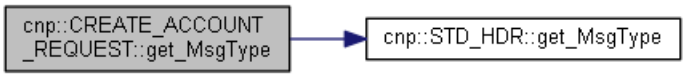
size_t cnp::CREATE_ACCOUNT_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

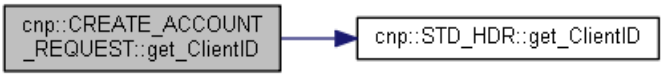
DWORD cnp::CREATE_ACCOUNT_REQUEST::get_MsgType (void) const

Here is the call graph for this function:



WORD cnp::CREATE_ACCOUNT_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

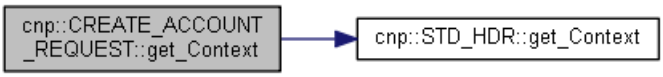


DWORD cp::CREATE_ACCOUNT_REQUEST::get_Sequence (void) const

Here is the call graph for this function:

DWORD cp::CREATE_ACCOUNT_REQUEST::get_Context (void) const

Here is the call graph for this function:



const char* cp::CREATE_ACCOUNT_REQUEST::get_FirstName (void) const

const char* cp::CREATE_ACCOUNT_REQUEST::get_LastName (void) const

const char* cp::CREATE_ACCOUNT_REQUEST::get_EmailAddress (void) const

WORD cp::CREATE_ACCOUNT_REQUEST::get_PIN (void) const

DWORD cp::CREATE_ACCOUNT_REQUEST::get_SSNumber (void) const

DWORD cp::CREATE_ACCOUNT_REQUEST::get_DLNumber (void) const

Member Data Documentation

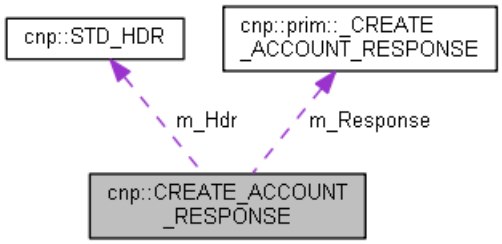
STD_HDR cp::CREATE_ACCOUNT_REQUEST::m_Hdr

prim::_CREATE_ACCOUNT_REQUEST cp::CREATE_ACCOUNT_REQUEST::m_Request

---

cp::CREATE_ACCOUNT_RESPONSE Struct Reference

[Server] Create Account Response message
Collaboration diagram for cp::CREATE_ACCOUNT_RESPONSE:



Public Member Functions

- CREATE_ACCOUNT_RESPONSE ()
  Default constructor.
- CREATE_ACCOUNT_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_CREATE_ACCOUNT_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19

See also:
cnp::CREATE_ACCOUNT_REQUEST

Constructor & Destructor Documentation

cnp::CREATE_ACCOUNT_RESPONSE::CREATE_ACCOUNT_RESPONSE ()

cnp::CREATE_ACCOUNT_RESPONSE::CREATE_ACCOUNT_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwSequence*, DWORD *dwContext*)

Parameters:

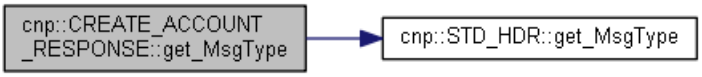
in	*dwResult*	Server generated cnp::CER_TYPE result

in	*wClientID*	Copied from CREATE_ACCOUNT_REQUEST
in	*dwSequence*	Copied from CREATE_ACCOUNT_REQUEST
in	*dwContext*	Copied from CREATE_ACCOUNT_REQUEST

Member Function Documentation

DWORD cnp::CREATE_ACCOUNT_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:



DWORD cnp::CREATE_ACCOUNT_RESPONSE::get_ResponseResult (void) const

size_t cnp::CREATE_ACCOUNT_RESPONSE::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

Member Data Documentation

STD_HDR cnp::CREATE_ACCOUNT_RESPONSE::m_Hdr

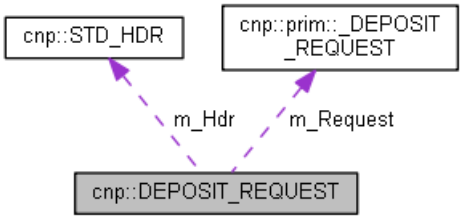
prim::_CREATE_ACCOUNT_RESPONSE cnp::CREATE_ACCOUNT_RESPONSE::m_Response

---

cnp::DEPOSIT_REQUEST Struct Reference

[Client] Deposit Request message

Collaboration diagram for cnp::DEPOSIT_REQUEST:



Public Member Functions

- DEPOSIT_REQUEST (WORD wClientID, DWORD dwAmount, DEPOSIT_TYPE Type, DWORD dwContext=0)

Initialization constructor.

- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- DWORD get_Amount (void) const
- WORD get_DepositType (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_DEPOSIT_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_dwAmount	16	19
m_Request	m_wType	20	21

Constructor & Destructor Documentation

cnp::DEPOSIT_REQUEST::DEPOSIT_REQUEST (WORD *wClientID*, DWORD *dwAmount*, DEPOSIT_TYPE *Type*, DWORD *dwContext* = 0)

Note:

auto increments the Client's global sequence number

Parameters:

wClientID	Server generated Client ID
dwAmount	Amount to deposit (in cents)
Type	cnp::DT_CASH or cnp::DT_CHECK
dwContext	[Optional] Client provided field

Member Function Documentation

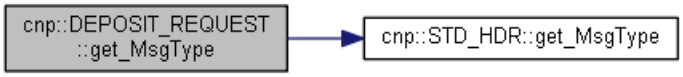
size_t cnp::DEPOSIT_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

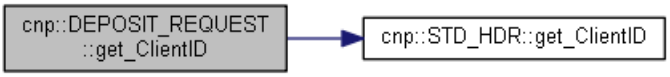
DWORD cnp::DEPOSIT_REQUEST::get_MsgType (void) const

Here is the call graph for this function:



WORD cnp::DEPOSIT_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

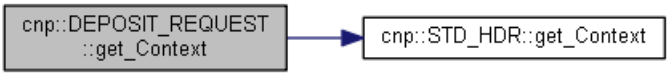


DWORD cnp::DEPOSIT_REQUEST::get_Sequence (void) const

Here is the call graph for this function:

DWORD cnp::DEPOSIT_REQUEST::get_Context (void) const

Here is the call graph for this function:



DWORD cnp::DEPOSIT_REQUEST::get_Amount (void) const

WORD cnp::DEPOSIT_REQUEST::get_DepositType (void) const

Member Data Documentation

STD_HDR cnp::DEPOSIT_REQUEST::m_Hdr

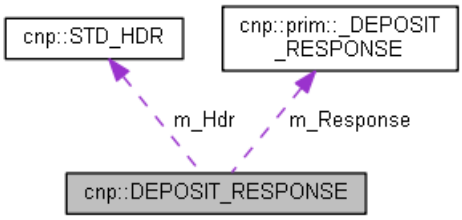
prim::_DEPOSIT_REQUEST cnp::DEPOSIT_REQUEST::m_Request

---

cnp::DEPOSIT_RESPONSE Struct Reference

[Server] Deposit Response message

Collaboration diagram for cnp::DEPOSIT_RESPONSE:



Public Member Functions

- DEPOSIT_RESPONSE ()
  Default Constructor.
- DEPOSIT_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_DEPOSIT_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19

See also:

cnp::DEPOSIT_REQUEST

Constructor & Destructor Documentation

cnp::DEPOSIT_RESPONSE::DEPOSIT_RESPONSE ()

cnp::DEPOSIT_RESPONSE::DEPOSIT_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwSequence*, DWORD *dwContext*)

Parameters:

dwResult	Server generated cnp::CER_TYPE result
wClientID	Copied from DEPOSIT_REQUEST
dwSequence	Copied from DEPOSIT_REQUEST
dwContext	Copied from DEPOSIT_REQUEST

Member Function Documentation

DWORD cnp::DEPOSIT_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:

DWORD cnp::DEPOSIT_RESPONSE::get_ResponseResult (void) const

size_t cnp::DEPOSIT_RESPONSE::get_Size (void) const

Member Data Documentation

STD_HDR cnp::DEPOSIT_RESPONSE::m_Hdr

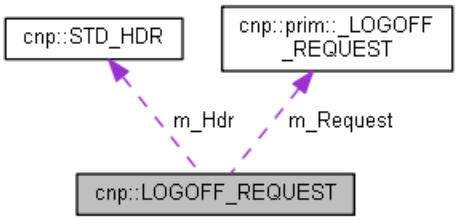
prim::_DEPOSIT_RESPONSE cnp::DEPOSIT_RESPONSE::m_Response

---

cnp::LOGOFF_REQUEST Struct Reference

[Client] Logoff Request message

Collaboration diagram for cnp::LOGOFF_REQUEST:



Public Member Functions

- LOGOFF_REQUEST ()
  Default Constructor.
- LOGOFF_REQUEST (WORD wClientID, DWORD dwContext=0)
  Initialization Constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_LOGOFF_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15

Constructor & Destructor Documentation

cnp::LOGOFF_REQUEST::LOGOFF_REQUEST ()

cnp::LOGOFF_REQUEST::LOGOFF_REQUEST (WORD *wClientID*, DWORD *dwContext* = 0)

Note:

auto increments the Client's global sequence number

Parameters:

wClientID	Server generated Client ID

dwContext	[Optional] Client provided field

Member Function Documentation

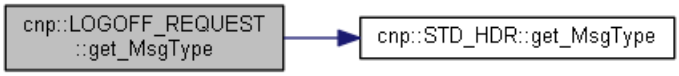
size_t cnp::LOGOFF_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

DWORD cnp::LOGOFF_REQUEST::get_MsgType (void) const

Here is the call graph for this function:

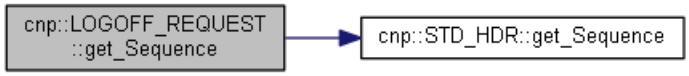


WORD cnp::LOGOFF_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

DWORD cnp::LOGOFF_REQUEST::get_Sequence (void) const

Here is the call graph for this function:



DWORD cnp::LOGOFF_REQUEST::get_Context (void) const

Here is the call graph for this function:

Member Data Documentation

STD_HDR cnp::LOGOFF_REQUEST::m_Hdr

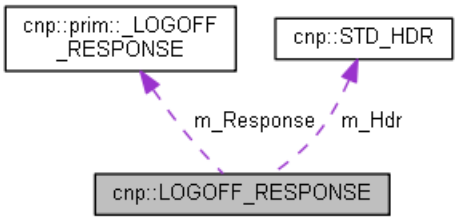
prim::_LOGOFF_REQUEST cnp::LOGOFF_REQUEST::m_Request

---

cnp::LOGOFF_RESPONSE Struct Reference

[Server] Logoff Response message

Collaboration diagram for cnp::LOGOFF_RESPONSE:



Public Member Functions

- LOGOFF_RESPONSE ()
  Default Constructor.
- LOGOFF_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_LOGOFF_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19

See also:

cnp::LOGOFF_REQUEST

Constructor & Destructor Documentation

cnp::LOGOFF_RESPONSE::LOGOFF_RESPONSE ()

cnp::LOGOFF_RESPONSE::LOGOFF_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwSequence*, DWORD *dwContext*)

Parameters:

dwResult	Server generated cnp::CER_TYPE result
wClientID	Copied from LOGOFF_REQUEST
dwSequence	Copied from LOGOFF_REQUEST
dwContext	Copied from LOGOFF_REQUEST

Member Function Documentation

DWORD cnp::LOGOFF_RESPONSE::get_MsgType (void) const

DWORD cnp::LOGOFF_RESPONSE::get_ResponseResult (void) const

size_t cnp::LOGOFF_RESPONSE::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

Member Data Documentation

STD_HDR cnp::LOGOFF_RESPONSE::m_Hdr

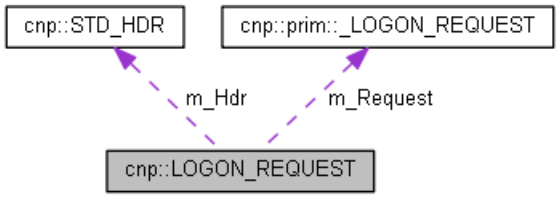
prim::_LOGOFF_RESPONSE cnp::LOGOFF_RESPONSE::m_Response

---

cnp::LOGON_REQUEST Struct Reference

[Client] Logon Request message

Collaboration diagram for cnp::LOGON_REQUEST:



Public Member Functions

- LOGON_REQUEST ()
  Default constructor.
- LOGON_REQUEST (WORD wClientID, const char *szFirstName, WORD wPIN, DWORD dwContext=0)
  Initialization constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- const char * get_FirstName (void) const
- WORD get_PIN (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_LOGON_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_szFirstName	16	47
m_Request	m_wPIN	48	49

Constructor & Destructor Documentation

cnp::LOGON_REQUEST::LOGON_REQUEST ()

cnp::LOGON_REQUEST::LOGON_REQUEST (WORD *wClientID*, const char * *szFirstName*, WORD *wPIN*, DWORD *dwContext* = 0)

Note:

auto increments the Client's global sequence number

Parameters:

wClientID	Server provided Client ID
szFirstName	Client field used in CREATE_ACCOUNT_REQUEST
wPIN	Client field used in CREATE_ACCOUNT_REQUEST
dwContext	[Optional] field provided for Client's use

Member Function Documentation

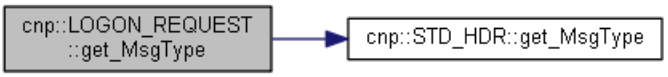
size_t cnp::LOGON_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

DWORD cnp::LOGON_REQUEST::get_MsgType (void) const

Here is the call graph for this function:

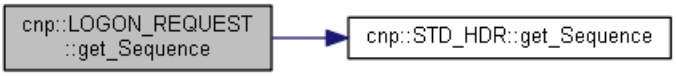


WORD cnp::LOGON_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

DWORD cnp::LOGON_REQUEST::get_Sequence (void) const

Here is the call graph for this function:



DWORD cnp::LOGON_REQUEST::get_Context (void) const

Here is the call graph for this function:

const char* cnp::LOGON_REQUEST::get_FirstName (void) const

WORD cnp::LOGON_REQUEST::get_PIN (void) const

Member Data Documentation

STD_HDR cnp::LOGON_REQUEST::m_Hdr

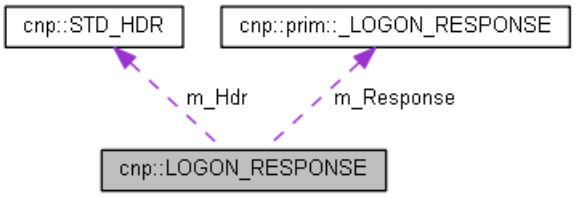
prim::_LOGON_REQUEST cnp::LOGON_REQUEST::m_Request

---

cnp::LOGON_RESPONSE Struct Reference

[Server] Logon Response message

Collaboration diagram for cnp::LOGON_RESPONSE:



Public Member Functions

- LOGON_RESPONSE ()
  Default Constructor.
- LOGON_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_LOGON_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11

m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19

See also:

cnp::LOGON_REQUEST

Constructor & Destructor Documentation

cnp::LOGON_RESPONSE::LOGON_RESPONSE ()

cnp::LOGON_RESPONSE::LOGON_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwSequence*, DWORD *dwContext*)

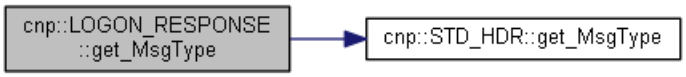
Parameters:

dwResult	Server generated cnp::CER_TYPE result
wClientID	Copied from LOGON_REQUEST
dwSequence	Copied from LOGON_REQUEST
dwContext	Copied from LOGON_REQUEST

Member Function Documentation

DWORD cnp::LOGON_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:



DWORD cnp::LOGON_RESPONSE::get_ResponseResult (void) const

size_t cnp::LOGON_RESPONSE::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

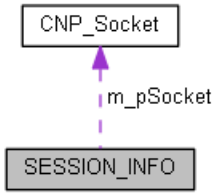
Member Data Documentation

STD_HDR cnp::LOGON_RESPONSE::m_Hdr

prim::_LOGON_RESPONSE cnp::LOGON_RESPONSE::m_Response

SESSION_INFO Struct Reference

Collaboration diagram for SESSION_INFO:



Public Types

- typedef cnp::WORD key_type

Public Member Functions

- SESSION_INFO (cnp::WORD wClientID, SESSION_STATE sState, CNP_Socket *pSocket=nullptr)
  Initialization Constructor.
- cnp::WORD get_ClientID (void) const throw ()
- void set_ClientID (cnp::WORD wSet) throw ()
- cnp::WORD get_State (void) const throw ()
- void set_State (SESSION_STATE sSet) throw ()
- const cnp::QWORD & get_CustomerID (void) const throw ()
- void set_CustomerID (const cnp::QWORD &qwSet) throw ()

Public Attributes

- cnp::WORD m_wClientID
  Key field.
- cnp::WORD m_wState
- CNP_Socket * m_pSocket
- cnp::QWORD m_qwCustomerID

Detailed Description

SESSION_INFO is a runtime only data-structure used to maintain an association between Client ID, session state, socket connection & Customer ID.

Member Typedef Documentation

typedef cnp::WORD SESSION_INFO::key_type

Constructor & Destructor Documentation

SESSION_INFO::SESSION_INFO (cnp::WORD *wClientID*, SESSION_STATE *sState*, CNP_Socket * *pSocket* = `nullptr`)

Member Function Documentation

cnp::WORD SESSION_INFO::get_ClientID (void) const throw)

void SESSION_INFO::set_ClientID (cnp::WORD *wSet*) throw)

cnp::WORD SESSION_INFO::get_State (void) const throw)

void SESSION_INFO::set_State (SESSION_STATE *sSet*) throw)

const cnp::QWORD& SESSION_INFO::get_CustomerID (void) const throw)

void SESSION_INFO::set_CustomerID (const cnp::QWORD & *qwSet*) throw)

Member Data Documentation

cnp::WORD SESSION_INFO::m_wClientID

cnp::WORD SESSION_INFO::m_wState

CNP_Socket* SESSION_INFO::m_pSocket

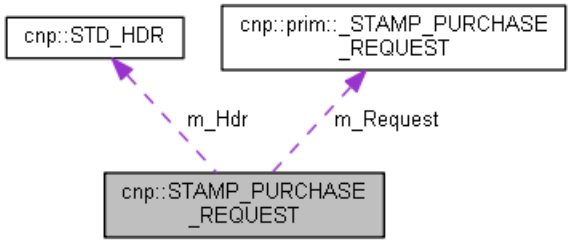
cnp::QWORD SESSION_INFO::m_qwCustomerID

---

cnp::STAMP_PURCHASE_REQUEST Struct Reference

[Client] Stamp Purchase Request Message

Collaboration diagram for cnp::STAMP_PURCHASE_REQUEST:



Public Member Functions

- STAMP_PURCHASE_REQUEST (WORD wClientID, DWORD dwAmount, DWORD dwContext=0)
  Initialization constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const

- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- DWORD get_Amount (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_STAMP_PURCHASE_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_dwAmount	16	19

Constructor & Destructor Documentation

cnp::STAMP_PURCHASE_REQUEST::STAMP_PURCHASE_REQUEST (WORD *wClientID*, DWORD *dwAmount*, DWORD *dwContext* = 0)

Parameters:

in	*wClientID*	Server generated Client ID
in	*dwAmount*	Cost of stamps attempting to purchase (in cents) (i.e. 1000 = $10.00)
in	*dwContext*	[Optional] field provided by the Client

Note:

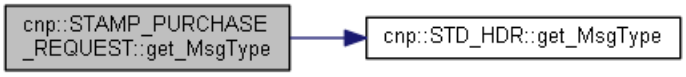
auto increments the Client's global sequence number

Member Function Documentation

size_t cnp::STAMP_PURCHASE_REQUEST::get_Size (void) const

DWORD cnp::STAMP_PURCHASE_REQUEST::get_MsgType (void) const

Here is the call graph for this function:

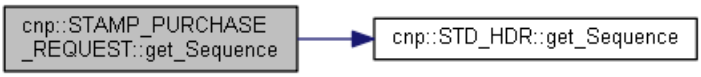


WORD cnp::STAMP_PURCHASE_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

DWORD cnp::STAMP_PURCHASE_REQUEST::get_Sequence (void) const

Here is the call graph for this function:



DWORD cnp::STAMP_PURCHASE_REQUEST::get_Context (void) const

Here is the call graph for this function:

DWORD cnp::STAMP_PURCHASE_REQUEST::get_Amount (void) const

Member Data Documentation

STD_HDR cnp::STAMP_PURCHASE_REQUEST::m_Hdr

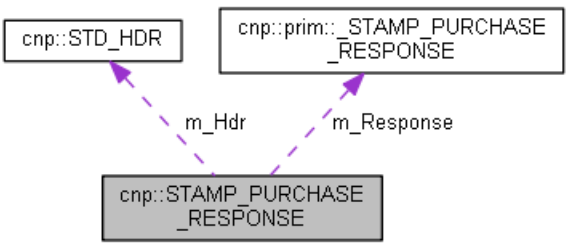
prim::_STAMP_PURCHASE_REQUEST cnp::STAMP_PURCHASE_REQUEST::m_Request

---

cnp::STAMP_PURCHASE_RESPONSE Struct Reference

[Server] Stamp Purchase Response message

Collaboration diagram for cnp::STAMP_PURCHASE_RESPONSE:



Public Member Functions

- STAMP_PURCHASE_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_STAMP_PURCHASE_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19

See also:
cnp::STAMP_PURCHASE_REQUEST

Constructor & Destructor Documentation

cnp::STAMP_PURCHASE_RESPONSE::STAMP_PURCHASE_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwSequence*, DWORD *dwContext*)

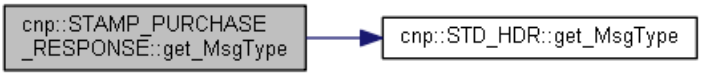
Parameters:

in	*dwResult*	Server generated cnp::CER_TYPE result
in	*wClientID*	Copied from STAMP_PURCHASE_REQUEST
in	*dwSequence*	Copied from STAMP_PURCHASE_REQUEST
in	*dwContext*	Copied from STAMP_PURCHASE_REQUEST

Member Function Documentation

DWORD cnp::STAMP_PURCHASE_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:



DWORD cnp::STAMP_PURCHASE_RESPONSE::get_ResponseResult (void) const

size_t cnp::STAMP_PURCHASE_RESPONSE::get_Size (void) const

Member Data Documentation

STD_HDR cnp::STAMP_PURCHASE_RESPONSE::m_Hdr

prim::_STAMP_PURCHASE_RESPONSE cnp::STAMP_PURCHASE_RESPONSE::m_Response

---

cnp::STD_HDR Struct Reference

CNP Standard Message Header.

Public Member Functions

- STD_HDR ()
  Default constructor.
- STD_HDR (DWORD dwMsgType, WORD wDataLen, WORD wClientID=INVALID_CLIENT_ID, DWORD dwSequence=0, DWORD dwContext=0)
  Initialization constructor.
- STD_HDR (const STD_HDR &rhs)
  Copy Constructor.
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const

Public Attributes

- DWORD m_dwMsgType
  Message Type.
- WORD m_wDataLen
  Message data length excluding this header.
- WORD m_wClientID
- DWORD m_dwSequence
- DWORD m_dwContext
  [Optional] field, reserved for the Client's use

Detailed Description

This is the header message that is provided as part of all CNP Request and Response messages. The m_wDataLen field is subsequently calculated as:

```
sizeof(m_Request) or sizeof(m_Response)
```

to get the size of message excluding the size of the header.

The m_dwContext field is exclusively reserved for application use. The contents of this field is returned to the user in the corresponding result message structure without modification.

The m_wClientID value is return by the server in the CONNECTION_RESPONSE message and is required in all subsequent messages sent by the client to the server.

See also:

CONNECT_RESPONSE

Field(s)	Begin Byte	End Byte
m_dwMsgType	0	3
m_wDataLen	4	5
m_wClientID	6	7
m_dwSequence	8	11
m_dwContext	12	15

Constructor & Destructor Documentation

cnp::STD_HDR::STD_HDR ()

cnp::STD_HDR::STD_HDR (DWORD *dwMsgType*, WORD *wDataLen*, WORD *wClientID* = INVALID CLIENT ID, DWORD *dwSequence* = 0, DWORD *dwContext* = 0)

Parameters:

in	*dwMsgType*	MSG_TYPE
in	*wDataLen*	Message data length excluding this header
in	*wClientID*	Client ID, initially set by the Server & used by Client in subsequent messages
in	*dwSequence*	Incremented by the Client, used to match Server responses to Client requests
in	*dwContext*	[Optional] field for Client's use

cnp::STD_HDR::STD_HDR (const STD_HDR & *rhs*)

Member Function Documentation

DWORD cnp::STD_HDR::get_MsgType (void) const

WORD cnp::STD_HDR::get_ClientID (void) const

DWORD cnp::STD_HDR::get_Sequence (void) const

DWORD cnp::STD_HDR::get_Context (void) const

Member Data Documentation

DWORD cnp::STD_HDR::m_dwMsgType

WORD cnp::STD_HDR::m_wDataLen

WORD cnp::STD_HDR::m_wClientID

Client ID, initially set by the Server & used by Client in subsequent messages

DWORD cnp::STD_HDR::m_dwSequence

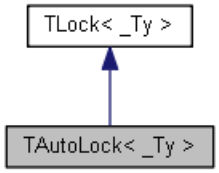
Incremented by the Client, used to match Server responses to Client requests

DWORD cnp::STD_HDR::m_dwContext

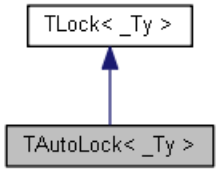
TAutoLock< _Ty > Class Template Reference

"Stack-based" Template

Inheritance diagram for TAutoLock< _Ty >:



Collaboration diagram for TAutoLock< _Ty >:



Public Member Functions

- TAutoLock (_Ty *pCS)
- ~TAutoLock ()

Protected Member Functions

- TAutoLock ()
- TAutoLock (const TAutoLock< _Ty > &o)

Detailed Description

template<class _Ty>class TAutoLock< _Ty >

TAutoLock encapsulates some of the mundane operations of its base class - TLock.

Is meant to be instantiated on the Stack so that on return of the 'Locked' function, it is Unlocked with the destruction of the object.

Warning:

This template class expects 'class _Ty' to support the Lock/Unlock methods

---

Constructor & Destructor Documentation

template<class _Ty> TAutoLock< _Ty >::TAutoLock () `[protected]`

Declare the Default and Copy Constructor protected so the class cannot be instantiated in this manner

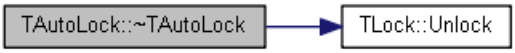
template<class _Ty> TAutoLock< _Ty >::TAutoLock (const TAutoLock< _Ty > & *o*) `[protected]`

template<class _Ty> TAutoLock< _Ty >::TAutoLock (_Ty * *pCS*) `[explicit]`

Here is the call graph for this function:

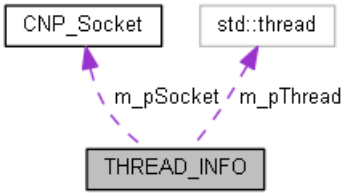
template<class _Ty> TAutoLock< _Ty >::~TAutoLock ()

Here is the call graph for this function:



THREAD_INFO Struct Reference

Collaboration diagram for THREAD_INFO:



Public Member Functions

- THREAD_INFO (void)

- ~THREAD_INFO (void)

Public Attributes

- std::atomic< bool > m_bTerminate
- CNP_Socket * m_pSocket
- std::thread * m_pThread

Private Member Functions

- THREAD_INFO (const THREAD_INFO &)
- THREAD_INFO & operator= (const THREAD_INFO &)

Detailed Description

A basic data-structure passed to the thread function

Constructor & Destructor Documentation

THREAD_INFO::THREAD_INFO (void)

THREAD_INFO::~THREAD_INFO (void)

THREAD_INFO::THREAD_INFO (const THREAD_INFO &)`[private]`

Member Function Documentation

THREAD_INFO& THREAD_INFO::operator= (const THREAD_INFO &)`[private]`

Member Data Documentation

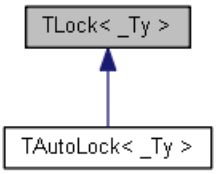
std::atomic<bool> THREAD_INFO::m_bTerminate

CNP_Socket* THREAD_INFO::m_pSocket

std::thread* THREAD_INFO::m_pThread

TLock< _Ty > Class Template Reference

"Stack-based" Template
Inheritance diagram for TLock< _Ty >:



Public Member Functions

- TLock ()
- void Lock (void)
- void Unlock (void)
- void SetLock (_Ty *pCS)

Protected Member Functions

- TLock (const TLock< _Ty > &)
- TLock (_Ty *pCS)

Private Attributes

- _Ty * m_pCS

Detailed Description

template<class _Ty>class TLock< _Ty >

This implementation does NOT actually contain a synchronization object, but is used to manipulate an existing one.

Warning:

This template class expects 'class _Ty' to support the Lock/Unlock methods

---

Constructor & Destructor Documentation

template<class _Ty > TLock< _Ty >::TLock (const TLock< _Ty > &) [protected]

Declare the Default and Copy Constructor protected so the class cannot be instantiated in this manner

template<class _Ty > TLock< _Ty >::TLock (_Ty * *pCS*) [protected]

template<class _Ty > TLock< _Ty >::TLock ()

Member Function Documentation

template<class _Ty > void TLock< _Ty >::Lock (void)

template<class _Ty > void TLock< _Ty >::Unlock (void)

template<class _Ty > void TLock< _Ty >::SetLock (_Ty * *pCS*)

Here is the call graph for this function:

Member Data Documentation

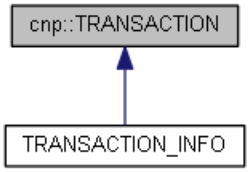
template<class _Ty > _Ty* TLock< _Ty >::m_pCS `[private]`

---

cnp::TRANSACTION Struct Reference

A Customer Transaction Record.

Inheritance diagram for cnp::TRANSACTION:



Public Member Functions

- TRANSACTION (void)
- TRANSACTION (DWORD dwID, const QWORD &qwDateTime, DWORD dwAmount, WORD wType)
- DWORD get_ID (void) const
- DWORD get_Amount (void) const
- const QWORD & get_DateTime (void) const
- TRANSACTION_TYPE get_Type (void) const

Public Attributes

- DWORD m_dwID
  A Server generated unique sequential ID associated with each transaction.
- QWORD m_qwDateTime
  a 64bit UTC value that represents number of seconds since Epoch
- DWORD m_dwAmount
  Amount excluding decimal point (i.e. $100.00 would be 10000)

- WORD m_wType
  The transaction type, represented as TT_DEPOSIT or TT_WITHDRAWAL.

Detailed Description

Field(s)	Begin Byte	End Byte
m_dwID	0	3
m_qwDateTime	4	11
m_dwAmount	12	15
m_wType	16	17

See also:
TRANSACTION_TYPE

Constructor & Destructor Documentation

cnp::TRANSACTION::TRANSACTION (void)

cnp::TRANSACTION::TRANSACTION (DWORD *dwID*, const QWORD & *qwDateTime*, DWORD *dwAmount*, WORD *wType*)

Member Function Documentation

DWORD cnp::TRANSACTION::get_ID (void) const

DWORD cnp::TRANSACTION::get_Amount (void) const

const QWORD& cnp::TRANSACTION::get_DateTime (void) const

TRANSACTION_TYPE cnp::TRANSACTION::get_Type (void) const

Member Data Documentation

DWORD cnp::TRANSACTION::m_dwID

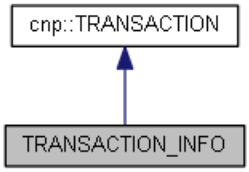
QWORD cnp::TRANSACTION::m_qwDateTime

DWORD cnp::TRANSACTION::m_dwAmount

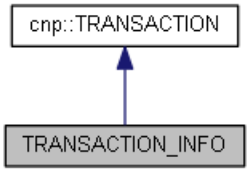
WORD cnp::TRANSACTION::m_wType

TRANSACTION_INFO Struct Reference

Inheritance diagram for TRANSACTION_INFO:



Collaboration diagram for TRANSACTION_INFO:



Public Types

- typedef cnp::DWORD key_type

Public Member Functions

- TRANSACTION_INFO (void)
  Default Constructor.
- TRANSACTION_INFO (cnp::DWORD dwID, const cnp::QWORD &qwDateTime, cnp::DWORD dwAmount, cnp::WORD wType, const cnp::QWORD &qwCustomerID)
  Initialization Constructor.
- TRANSACTION_INFO (const TRANSACTION_INFO &rhs)
  Copy Constructor.
- cnp::DWORD get_PrimaryKey (void) const throw ()
- const cnp::QWORD & get_CustomerID (void) const throw ()

Public Attributes

- cnp::QWORD m_qwCustomerID

Detailed Description

TRANSACTION_INFO is used to maintain a listing of all transactions related to a specific customer. The Transaction ID is used as the primary key.

Member Typedef Documentation

typedef cnp::DWORD TRANSACTION_INFO::key_type

Constructor & Destructor Documentation

TRANSACTION_INFO::TRANSACTION_INFO (void)

TRANSACTION_INFO::TRANSACTION_INFO (cnp::DWORD *dwID*, const cnp::QWORD & *qwDateTime*, cnp::DWORD *dwAmount*, cnp::WORD *wType*, const cnp::QWORD & *qwCustomerID*)

Parameters:

in	*dwID*	A Server generated unique sequential ID associated with each transaction
in	*qwDateTime*	64bit UTC value that represents number of seconds since Epoch
in	*dwAmount*	Amount excluding decimal point (i.e. $100.00 would be 10000)
in	*wType*	The transaction type, represented as cnp::TT_DEPOSIT or cnp::TT_WITHDRAWAL
in	*qwCustomerID*	Unique customer ID associated with the transaction

TRANSACTION_INFO::TRANSACTION_INFO (const TRANSACTION_INFO & *rhs*)

Member Function Documentation

cnp::DWORD TRANSACTION_INFO::get_PrimaryKey (void) const throw)

This method is used to provide a generic interface to retrieve TRANSACTION_INFO's primary key field. In this implementation, it is a wrapper around cnp::TRANSACTION.get_ID().

Return values:

cnp::DWORD	containing the primary key value

Here is the call graph for this function:

const cnp::QWORD& TRANSACTION_INFO::get_CustomerID (void) const throw)

Member Data Documentation

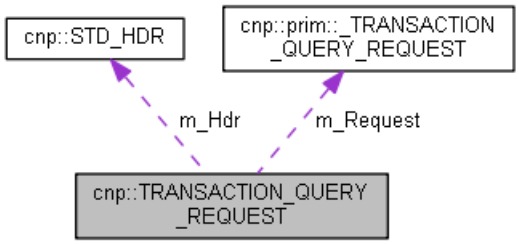
cnp::QWORD TRANSACTION_INFO::m_qwCustomerID

---

cnp::TRANSACTION_QUERY_REQUEST Struct Reference

[Client] Transaction Query Request message

Collaboration diagram for cnp::TRANSACTION_QUERY_REQUEST:



Public Member Functions

- TRANSACTION_QUERY_REQUEST (WORD wClientID, DWORD dwStartID, WORD wTransactionCount, DWORD dwContext=0)
  Initialization constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- DWORD get_StartID (void) const
- WORD get_TransactionCount (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_TRANSACTION_QUERY_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_dwStartID	16	19
m_Request	m_wTransactionCount	20	21

Constructor & Destructor Documentation

cnp::TRANSACTION_QUERY_REQUEST::TRANSACTION_QUERY_REQUEST (WORD *wClientID*, DWORD *dwStartID*, WORD *wTransactionCount*, DWORD *dwContext* = 0)

Note:

auto increments the Client's global sequence number

Parameters:

wClientID	Server generated Client ID
dwStartID	Transaction Record ID to begin query from
wTransactionCount	Number of Records requested
dwContext	[Optional] Client provided field

Member Function Documentation

size_t cnp::TRANSACTION_QUERY_REQUEST::get_Size (void) const

Return values:

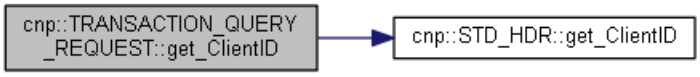
size_t	containing the size of the message in bytes

DWORD cnp::TRANSACTION_QUERY_REQUEST::get_MsgType (void) const

Here is the call graph for this function:

WORD cnp::TRANSACTION_QUERY_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

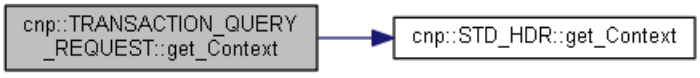


DWORD cnp::TRANSACTION_QUERY_REQUEST::get_Sequence (void) const

Here is the call graph for this function:

DWORD cnp::TRANSACTION_QUERY_REQUEST::get_Context (void) const

Here is the call graph for this function:



DWORD cnp::TRANSACTION_QUERY_REQUEST::get_StartID (void) const

WORD cnp::TRANSACTION_QUERY_REQUEST::get_TransactionCount (void) const

---

Member Data Documentation

STD_HDR cnp::TRANSACTION_QUERY_REQUEST::m_Hdr

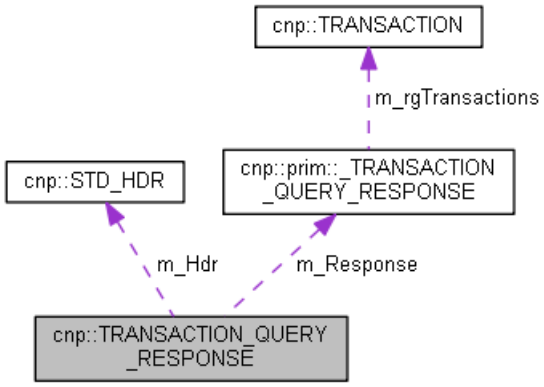
prim::_TRANSACTION_QUERY_REQUEST cnp::TRANSACTION_QUERY_REQUEST::m_Request

---

cnp::TRANSACTION_QUERY_RESPONSE Struct Reference

[Server] Transaction Query Response message

Collaboration diagram for cnp::TRANSACTION_QUERY_RESPONSE:



Public Member Functions

- TRANSACTION_QUERY_RESPONSE (DWORD dwResult, WORD wClientID, WORD wTransactionCount, DWORD dwSequence, DWORD dwContext)
  Initialization Constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- WORD get_TransactionCount (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_TRANSACTION_QUERY_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19
m_Response	m_wTransactionCount	20	21
m_Response	m_rgTransactions[]	22	...

See also:

cnp::TRANSACTION_QUERY_REQUEST
cnp::TRANSACTION

Constructor & Destructor Documentation

cnp::TRANSACTION_QUERY_RESPONSE::TRANSACTION_QUERY_RESPONSE (DWORD *dwResult*, WORD *wClientID*, WORD *wTransactionCount*, DWORD *dwSequence*, DWORD *dwContext*)

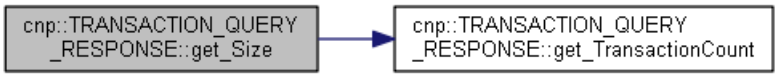
Parameters:

dwResult	Server generated cnp::CER_TYPE result
wClientID	Copied from TRANSACTION_QUERY_REQUEST
wTransactionCount	Actual number of records returned
dwSequence	Copied from TRANSACTION_QUERY_REQUEST
dwContext	Copied from TRANSACTION_QUERY_REQUEST

Member Function Documentation

size_t cnp::TRANSACTION_QUERY_RESPONSE::get_Size (void) const

Here is the call graph for this function:



DWORD cnp::TRANSACTION_QUERY_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:

DWORD cnp::TRANSACTION_QUERY_RESPONSE::get_ResponseResult (void) const

WORD cnp::TRANSACTION_QUERY_RESPONSE::get_TransactionCount (void) const

---

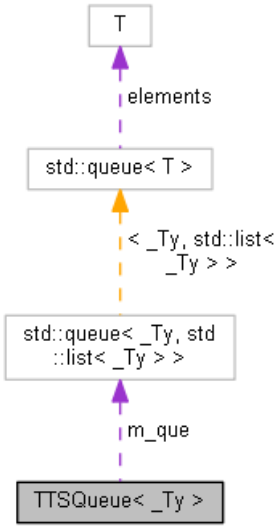
Member Data Documentation

STD_HDR cnp::TRANSACTION_QUERY_RESPONSE::m_Hdr

prim::_TRANSACTION_QUERY_RESPONSE
cnp::TRANSACTION_QUERY_RESPONSE::m_Response

TTSQueue< _Ty > Class Template Reference

Collaboration diagram for TTSQueue< _Ty >:



Public Member Functions

- TTSQueue ()
- TTSQueue (const _Myt &o)
- _Myt & operator= (const _Myt &rhs)
- virtual ~TTSQueue ()
- void Lock (void) const
- void Unlock (void) const
- void Init (bool bLock=true)
  Initializes the que by popping all elements until empty.
- bool GetItems (items_type &Items, bool bLock=true) const
  Returns a snapshot of queued-items. return == true if queue is not empty.
- const que_type & GetItems (void) const
  Returns a const reference to private stl::queue.

- const _Ty & Front (void) const
- void Push (const _Ty &Item, bool bLock=true)
  Adds an element to the back of the queue.
- bool Pop (bool bLock=true)
- bool PopFront (_Ty &Item, bool bLock=true)
- bool IsEmpty (void) const
  Tests if a queue is empty.
- size_type Size (void) const
  Returns the number of elements in the queue.

Private Types

- typedef std::queue< _Ty, std::list< _Ty > > que_type
- typedef que_type::size_type size_type
- typedef std::vector< _Ty > items_type
- typedef TTSQueue< _Ty > _Myt
- typedef TLock< _Myt > _Lock
- typedef TAutoLock< _Myt > _AutoLock

Private Attributes

- CAutoCriticalSection m_cs
- que_type m_que

Member Typedef Documentation

template<class _Ty > typedef std::queue<_Ty, std::list<_Ty> > TTSQueue< _Ty >::que_type [private]

template<class _Ty > typedef que_type::size_type TTSQueue< _Ty >::size_type [private]

template<class _Ty > typedef std::vector<_Ty> TTSQueue< _Ty >::items_type [private]

template<class _Ty > typedef TTSQueue<_Ty> TTSQueue< _Ty >::_Myt [private]

template<class _Ty > typedef TLock<_Myt> TTSQueue< _Ty >::_Lock [private]

template<class _Ty > typedef TAutoLock<_Myt> TTSQueue< _Ty >::_AutoLock [private]

Constructor & Destructor Documentation

template<class _Ty > TTSQueue< _Ty >::TTSQueue ()

template<class _Ty > TTSQueue< _Ty >::TTSQueue (const _Myt & *o*)

template<class _Ty > virtual TTSQueue< _Ty >::~TTSQueue () [virtual]

Member Function Documentation

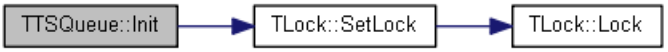
template<class _Ty > _Myt& TTSQueue< _Ty >::operator= (const _Myt & *rhs*)

template<class _Ty > void TTSQueue< _Ty >::Lock (void) const

template<class _Ty > void TTSQueue< _Ty >::Unlock (void) const

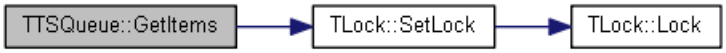
template<class _Ty > void TTSQueue< _Ty >::Init (bool *bLock* = `true`)

Here is the call graph for this function:



template<class _Ty > bool TTSQueue< _Ty >::GetItems (items_type & *Items*, bool *bLock* = `true`) const

Here is the call graph for this function:



template<class _Ty > const que_type& TTSQueue< _Ty >::GetItems (void) const

template<class _Ty > const _Ty& TTSQueue< _Ty >::Front (void) const

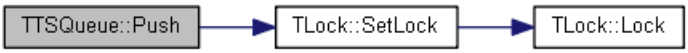
Returns a const reference to the first element at the front of the queue.

Warning:

If the queue is empty, the return value is undefined.

template<class _Ty > void TTSQueue< _Ty >::Push (const _Ty & *Item*, bool *bLock* = `true`)

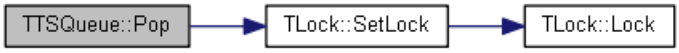
Here is the call graph for this function:



template<class _Ty > bool TTSQueue< _Ty >::Pop (bool *bLock* = `true`)

Removes an element from the front of the queue. Returns true if an element was removed.

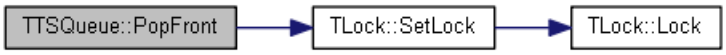
Here is the call graph for this function:



template<class _Ty > bool TTSQueue< _Ty >::PopFront (_Ty & *Item*, bool *bLock* = `true`)

Combines the operations of Pop & Front by returning a copy of the 'popped' element.

Here is the call graph for this function:



template<class _Ty > bool TTSQueue< _Ty >::IsEmpty (void) const

template<class _Ty > size_type TTSQueue< _Ty >::Size (void) const

---

Member Data Documentation

template<class _Ty > CAutoCriticalSection TTSQueue< _Ty >::m_cs[mutable], [private]

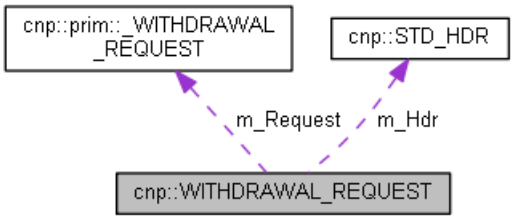
template<class _Ty > que_type TTSQueue< _Ty >::m_que[private]

---

cnp::WITHDRAWAL_REQUEST Struct Reference

[Client] Withdrawal Request message

Collaboration diagram for cnp::WITHDRAWAL_REQUEST:



Public Member Functions

- WITHDRAWAL_REQUEST (WORD wClientID, DWORD dwAmount, DWORD dwContext=0)
  Initialization constructor.
- size_t get_Size (void) const
- DWORD get_MsgType (void) const
- WORD get_ClientID (void) const
- DWORD get_Sequence (void) const
- DWORD get_Context (void) const
- DWORD get_Amount (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_WITHDRAWAL_REQUEST m_Request

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Request	m_dwAmount	16	19

Constructor & Destructor Documentation

cnp::WITHDRAWAL_REQUEST::WITHDRAWAL_REQUEST (WORD *wClientID*, DWORD *dwAmount*, DWORD *dwContext* = 0)

Note:

auto increments the Client's global sequence number

Parameters:

wClientID	Server generated Client ID
dwAmount	Amount Client wants to withdraw (in cents)
dwContext	[Optional] Client provided field

Member Function Documentation

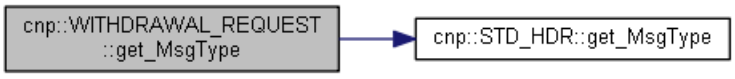
size_t cnp::WITHDRAWAL_REQUEST::get_Size (void) const

Return values:

size_t	containing the size of the message in bytes

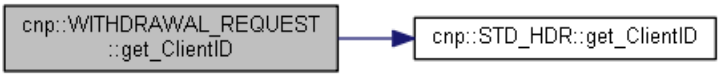
DWORD cnp::WITHDRAWAL_REQUEST::get_MsgType (void) const

Here is the call graph for this function:



WORD cnp::WITHDRAWAL_REQUEST::get_ClientID (void) const

Here is the call graph for this function:

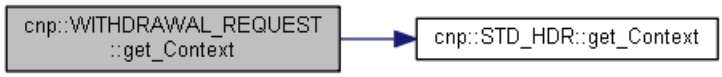


DWORD cnp::WITHDRAWAL_REQUEST::get_Sequence (void) const

Here is the call graph for this function:

DWORD cnp::WITHDRAWAL_REQUEST::get_Context (void) const

Here is the call graph for this function:



DWORD cnp::WITHDRAWAL_REQUEST::get_Amount (void) const

Member Data Documentation

STD_HDR cnp::WITHDRAWAL_REQUEST::m_Hdr

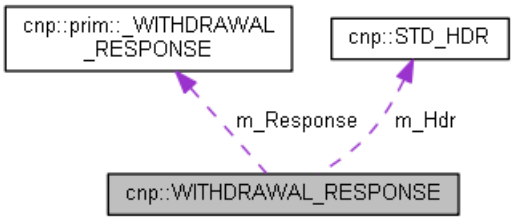
prim::_WITHDRAWAL_REQUEST cnp::WITHDRAWAL_REQUEST::m_Request

---

cnp::WITHDRAWAL_RESPONSE Struct Reference

[Server] Withdrawal Response message

Collaboration diagram for cnp::WITHDRAWAL_RESPONSE:



Public Member Functions

- WITHDRAWAL_RESPONSE (DWORD dwResult, WORD wClientID, DWORD dwSequence, DWORD dwContext)
- DWORD get_MsgType (void) const
- DWORD get_ResponseResult (void) const
- size_t get_Size (void) const

Public Attributes

- STD_HDR m_Hdr
- prim::_WITHDRAWAL_RESPONSE m_Response

Detailed Description

Message Members	Field	Begin Byte	End Byte
m_Hdr	m_dwMsgType	0	3
m_Hdr	m_wDataLen	4	5
m_Hdr	m_wClientID	6	7
m_Hdr	m_dwSequence	8	11
m_Hdr	m_dwContext	12	15
m_Response	m_dwResult	16	19

See also:

cnp::WITHDRAWAL_REQUEST

Constructor & Destructor Documentation

cnp::WITHDRAWAL_RESPONSE::WITHDRAWAL_RESPONSE (DWORD *dwResult*, WORD *wClientID*, DWORD *dwSequence*, DWORD *dwContext*)

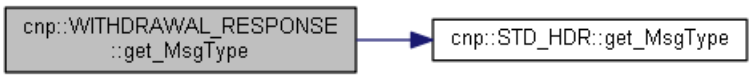
Parameters:

dwResult	Server generated cnp::CER_TYPE result
wClientID	Copied from WITHDRAWAL_REQUEST
dwSequence	Copied from WITHDRAWAL_REQUEST
dwContext	Copied from WITHDRAWAL_REQUEST

Member Function Documentation

DWORD cnp::WITHDRAWAL_RESPONSE::get_MsgType (void) const

Here is the call graph for this function:



DWORD cnp::WITHDRAWAL_RESPONSE::get_ResponseResult (void) const

size_t cnp::WITHDRAWAL_RESPONSE::get_Size (void) const

Member Data Documentation

STD_HDR cnp::WITHDRAWAL_RESPONSE::m_Hdr

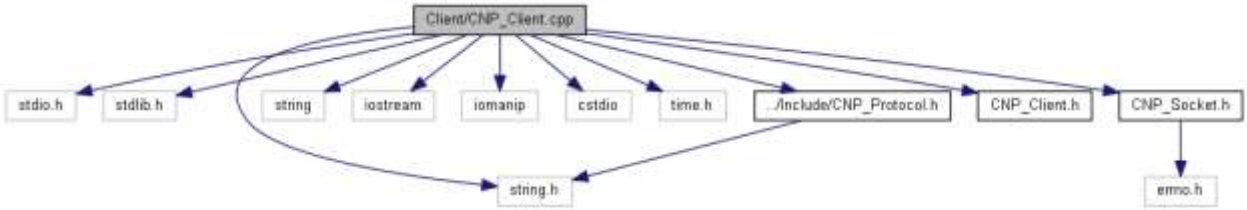
prim::_WITHDRAWAL_RESPONSE cnp::WITHDRAWAL_RESPONSE::m_Response

---

File Documentation

Client/CNP_Client.cpp File Reference

Include dependency graph for CNP_Client.cpp:



Macros

- #define CASE_CERTYPE(cer)

Functions

- const char * CerTypeToString (cnp::CER_TYPE cerType)
- const char * TransTypeToString (cnp::TRANSACTION_TYPE tType)
- std::string & RawTimeToLocalTimeString (time_t tRawTime, std::string &strLocalTime)
- void PrintBankMenu (void)
- cnp::CER_TYPE SendConnect (CNP_Socket &socket, cnp::WORD &wClientID)
- cnp::CER_TYPE SendCreateAccount (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendLogIn (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendLogOut (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendDeposit (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendWithdrawal (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendBalance (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendTransaction (CNP_Socket &socket, cnp::WORD wClientID)
- cnp::CER_TYPE SendStampPurchase (CNP_Socket &socket, cnp::WORD wClientID)
- int main (int argc, char *argv[])

Variables

- char g_szBuffer [512] = { 0 }

Detailed Description

Author:

Mark L. Short

Date:
April 10, 2015

Macro Definition Documentation

#define CASE_CERTYPE(cer)

```
Value:case cer: \
            return (const char*)#cer;
```

Function Documentation

const char* CerTypeToString (cnp::CER_TYPE *cerType*)

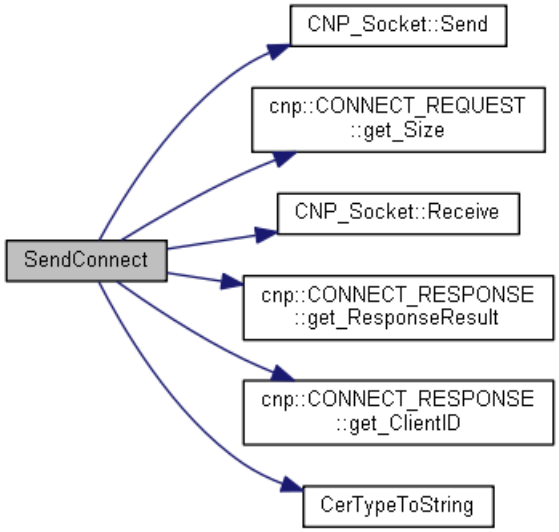
const char* TransTypeToString (cnp::TRANSACTION_TYPE *tType*)

std::string& RawTimeToLocalTimeString (time_t *tRawTime*, std::string & *strLocalTime*)

void PrintBankMenu (void)

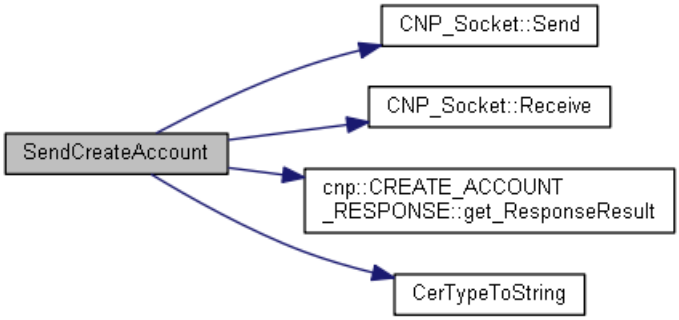
cnp::CER_TYPE SendConnect (CNP_Socket & *socket*, cnp::WORD & *wClientID*)

Here is the call graph for this function:



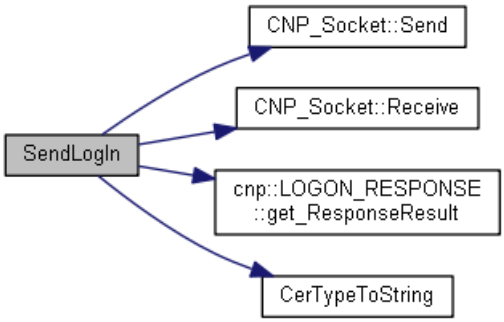
cnp::CER_TYPE SendCreateAccount (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



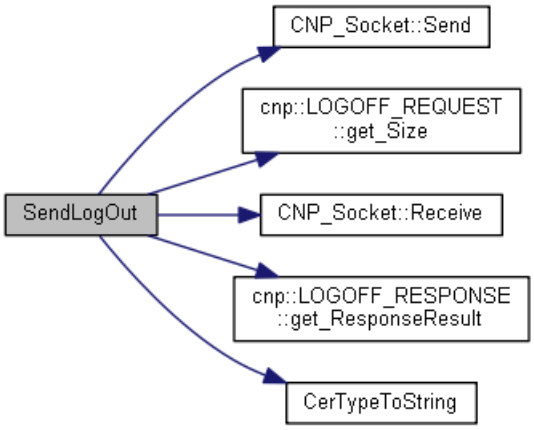
cnp::CER_TYPE SendLogIn (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



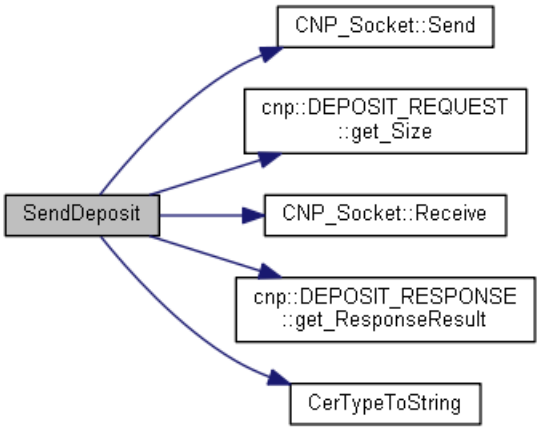
cnp::CER_TYPE SendLogOut (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



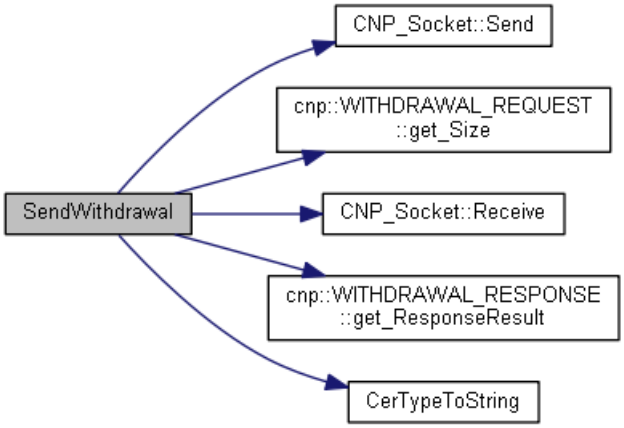
cnp::CER_TYPE SendDeposit (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



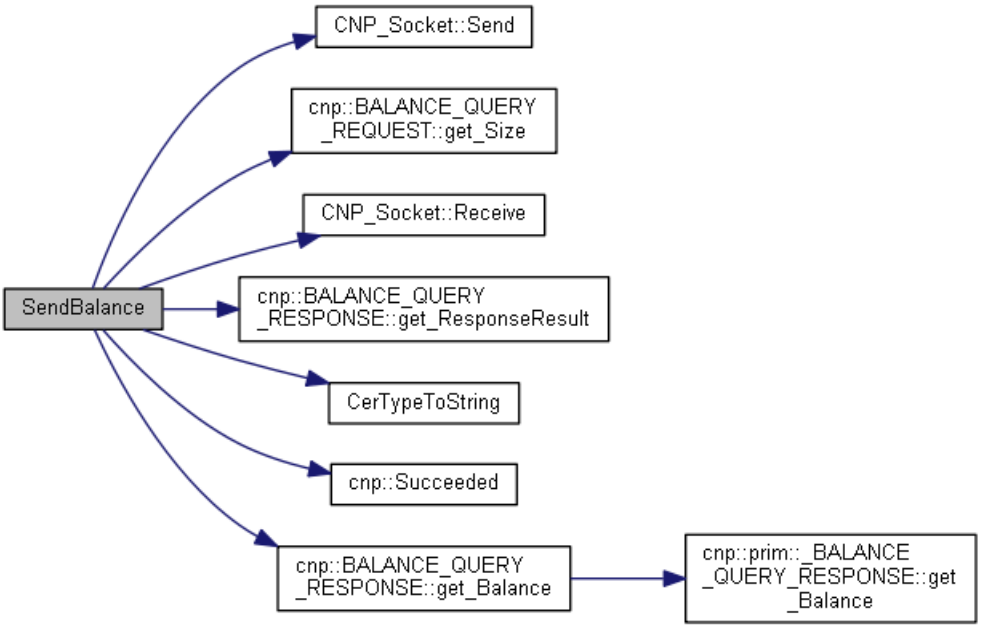
cnp::CER_TYPE SendWithdrawal (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



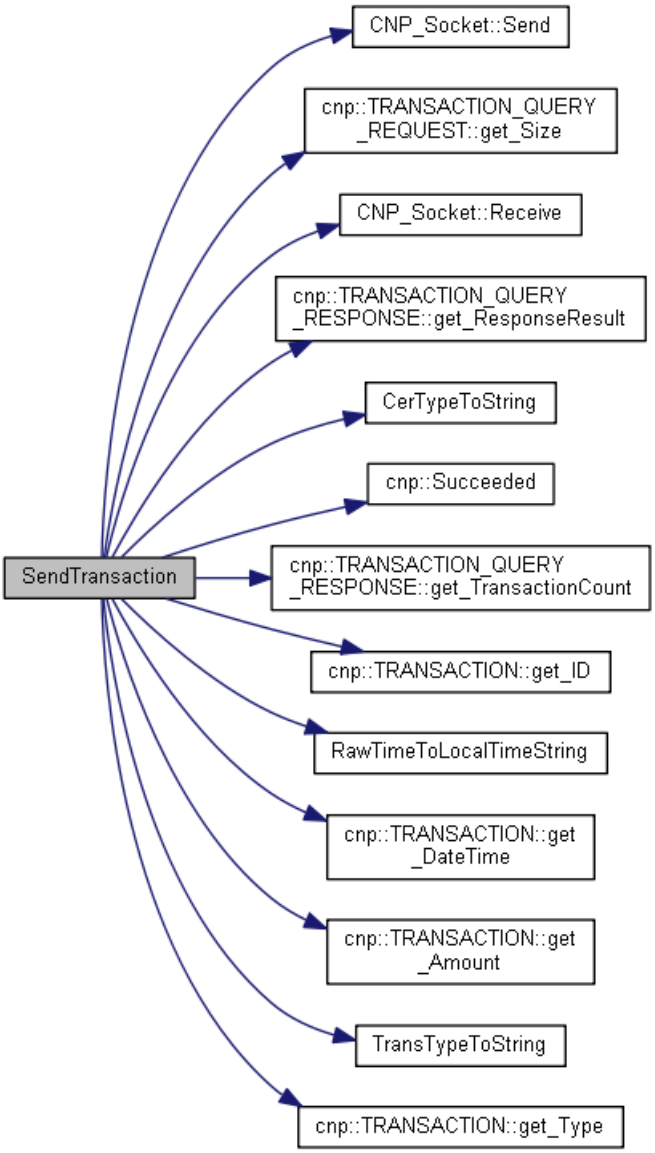
cnp::CER_TYPE SendBalance (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



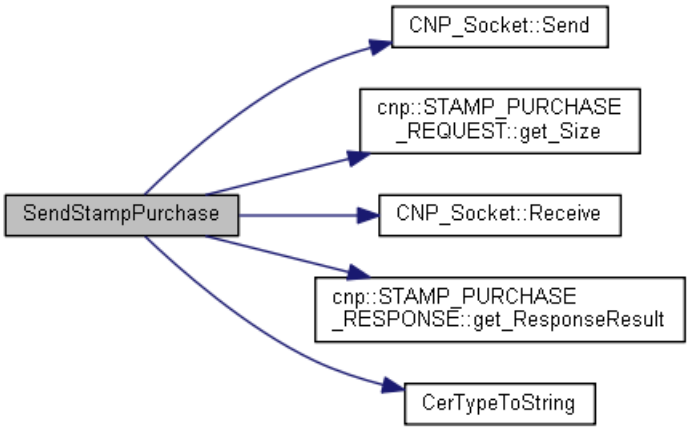
cnp::CER_TYPE SendTransaction (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



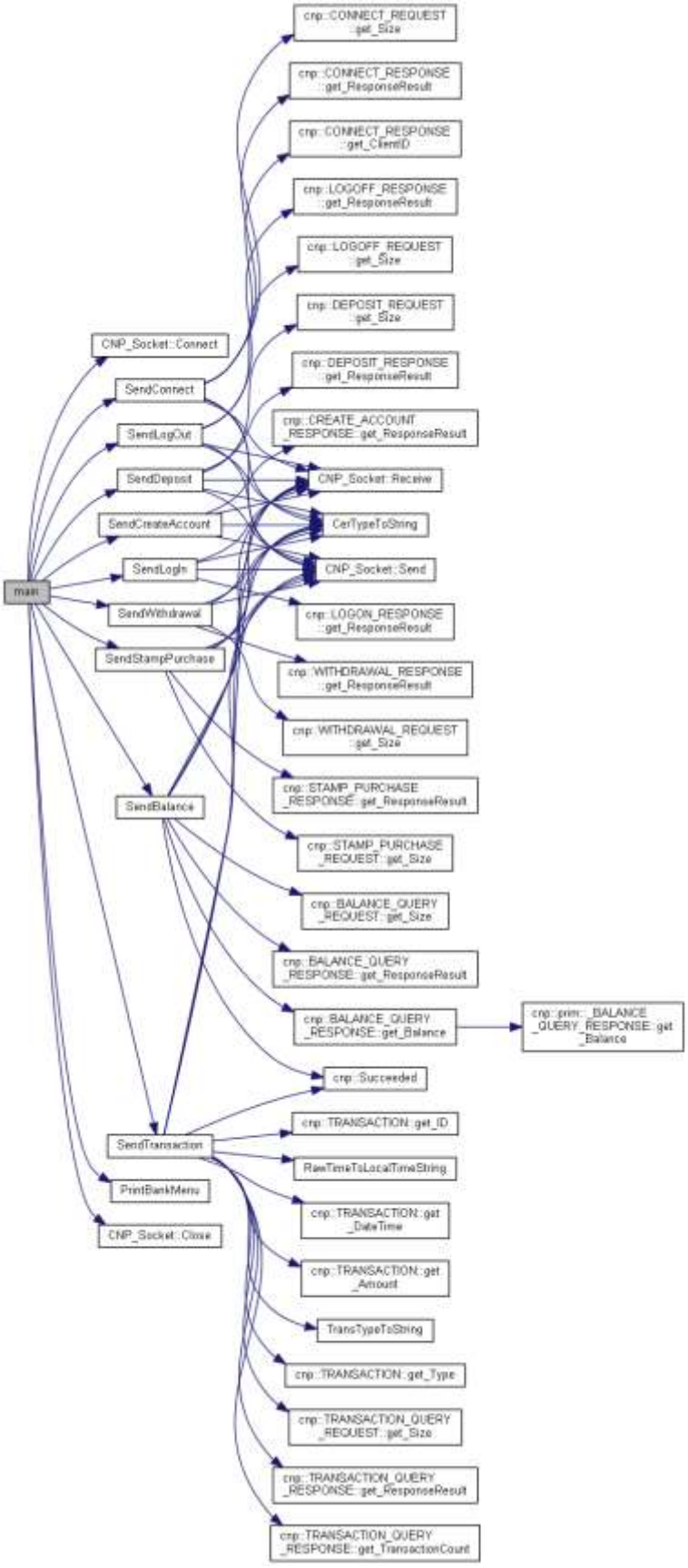
cnp::CER_TYPE SendStampPurchase (CNP_Socket & *socket*, cnp::WORD *wClientID*)

Here is the call graph for this function:



int main (int *argc*, char * *argv*[])

Here is the call graph for this function:


cnp::CONNECT_REQUEST::get_Size
cnp::CONNECT_RESPONSE::get_ResponseResult
cnp::CONNECT_RESPONSE::get_ClientID
cnp::LOGOFF_RESPONSE::get_ResponseResult
cnp::LOGOFF_REQUEST::get_Size
cnp::DEPOSIT_REQUEST::get_Size
CNP_Socket::Connect
cnp::DEPOSIT_RESPONSE::get_ResponseResult
SendConnect
cnp::CREATE_ACCOUNT_RESPONSE::get_ResponseResult
SendLogOut
SendDeposit
CNP_Socket::Receive
SendCreateAccount
CerTypeToString
SendLogIn
CNP_Socket::Send
main
SendWithdrawal
cnp::LOGON_RESPONSE::get_ResponseResult
SendStampPurchase
cnp::WITHDRAWAL_RESPONSE::get_ResponseResult
cnp::WITHDRAWAL_REQUEST::get_Size
cnp::STAMP_PURCHASE_RESPONSE::get_ResponseResult
SendBalance
cnp::STAMP_PURCHASE_REQUEST::get_Size
cnp::BALANCE_QUERY_REQUEST::get_Size
cnp::BALANCE_QUERY_RESPONSE::get_ResponseResult
cnp::BALANCE_QUERY_RESPONSE::get_Balance
cnp::prim::_BALANCE_QUERY_RESPONSE::get_Balance
cnp::Succeeded
cnp::TRANSACTION::get_ID
SendTransaction
RawTimeToLocalTimeString
PrintBankMenu
cnp::TRANSACTION::get_DateTime
CNP_Socket::Close
cnp::TRANSACTION::get_Amount
TransTypeToString
cnp::TRANSACTION::get_Type
cnp::TRANSACTION_QUERY_REQUEST::get_Size
cnp::TRANSACTION_QUERY_RESPONSE::get_ResponseResult
cnp::TRANSACTION_QUERY_RESPONSE::get_TransactionCount

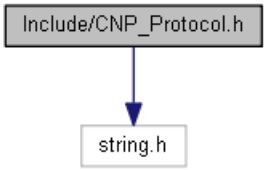
Variable Documentation

char g_szBuffer[512] = { 0 }

Client/CNP_Client.h File Reference

Include/CNP_Protocol.h File Reference

Contains type definitions required to support 5580 Computer Networks Project (CNP) Protocol.
Include dependency graph for CNP_Protocol.h:



Classes

- struct cnp::TRANSACTION
- *A Customer Transaction Record.* struct cnp::prim::_CONNECT_REQUEST
- *Connect Request Primitive.* struct cnp::prim::_CONNECT_RESPONSE
- *Connection Response Primitive.* struct cnp::prim::_CREATE_ACCOUNT_REQUEST
- *Create Account Request Primitive.* struct cnp::prim::_CREATE_ACCOUNT_RESPONSE
- *Create Account Response Primitive.* struct cnp::prim::_LOGON_REQUEST
- *Logon Request Primitive.* struct cnp::prim::_LOGON_RESPONSE
- *Logon Response Primitive.* struct cnp::prim::_LOGOFF_REQUEST
- *Logoff Request Primitive.* struct cnp::prim::_LOGOFF_RESPONSE
- *Logoff Response Primitive.* struct cnp::prim::_DEPOSIT_REQUEST
- *Deposit Request Primitive.* struct cnp::prim::_DEPOSIT_RESPONSE
- *Deposit Response Primitive.* struct cnp::prim::_WITHDRAWAL_REQUEST
- *Withdrawal Request Primitive.* struct cnp::prim::_WITHDRAWAL_RESPONSE
- *Withdrawal Response Primitive.* struct cnp::prim::_STAMP_PURCHASE_REQUEST
- *Purchase Stamp Request Primitive.* struct cnp::prim::_STAMP_PURCHASE_RESPONSE
- *Stamp Purchase Response Primitive.* struct cnp::prim::_TRANSACTION_QUERY_REQUEST
- *Transaction Query Request Primitive.* struct cnp::prim::_TRANSACTION_QUERY_RESPONSE
- *Transaction Query Result Primitive.* struct cnp::prim::_BALANCE_QUERY_REQUEST
- *Balance Query Request Primitive.* struct cnp::prim::_BALANCE_QUERY_RESPONSE
- *Balance Query Response Primitive.* struct cnp::STD_HDR
- *CNP Standard Message Header.* struct cnp::CONNECT_REQUEST
- *[Client] Connect Request message* struct cnp::CONNECT_RESPONSE

- *[Server] Connect Response message* struct cnp::CREATE_ACCOUNT_REQUEST
- *[Client] Create Account Request message* struct cnp::CREATE_ACCOUNT_RESPONSE
- *[Server] Create Account Response message* struct cnp::LOGON_REQUEST
- *[Client] Logon Request message* struct cnp::LOGON_RESPONSE
- *[Server] Logon Response message* struct cnp::LOGOFF_REQUEST
- *[Client] Logoff Request message* struct cnp::LOGOFF_RESPONSE
- *[Server] Logoff Response message* struct cnp::DEPOSIT_REQUEST
- *[Client] Deposit Request message* struct cnp::DEPOSIT_RESPONSE
- *[Server] Deposit Response message* struct cnp::WITHDRAWAL_REQUEST
- *[Client] Withdrawal Request message* struct cnp::WITHDRAWAL_RESPONSE
- *[Server] Withdrawal Response message* struct cnp::BALANCE_QUERY_REQUEST
- *[Client] Balance Query Request message* struct cnp::BALANCE_QUERY_RESPONSE
- *[Server] Balance Query Response message* struct cnp::TRANSACTION_QUERY_REQUEST
- *[Client] Transaction Query Request message* struct cnp::TRANSACTION_QUERY_RESPONSE
- *[Server] Transaction Query Response message* struct cnp::STAMP_PURCHASE_REQUEST
- *[Client] Stamp Purchase Request Message* struct cnp::STAMP_PURCHASE_RESPONSE

[Server] Stamp Purchase Response message Namespaces

- cnp
- cnp::prim

Macros

- #define COUNTOF(_array) (sizeof(_array) / sizeof(_array[0]))
  Helper macro that calculates count of elements in an array.
- #define MAKE_MSG_TYPE(type, sub) ((sub << 16) + type)
- #define MAKE_ERROR_RESULT(facility, sub) ((facility << 16) + sub)

Typedefs

- typedef unsigned short cnp::WORD
  16bit type definition
- typedef unsigned long cnp::DWORD
  32bit type definition
- typedef unsigned long long cnp::QWORD
  64bit type definition

Enumerations

- enum cnp::CNP_MSG_TYPE { cnp::CMT_INVALID = 0x00, cnp::CMT_CONNECT = 0x50, cnp::CMT_CREATE_ACCOUNT = 0x51, cnp::CMT_LOGON = 0x52, cnp::CMT_LOGOFF = 0x53, cnp::CMT_DEPOSIT = 0x54, cnp::CMT_WITHDRAWAL = 0x55, cnp::CMT_BALANCE_QUERY = 0x56, cnp::CMT_TRANSACTION_QUERY = 0x57, cnp::CMT_PURCHASE_STAMPS = 0x58 }
- *Supported CNP Message Types (CMT_)* enum cnp::CNP_MSG_SUBTYPE { cnp::CMS_INVALID = 0x00, cnp::CMS_REQUEST = 0x01, cnp::CMS_RESPONSE = 0x02 }
- *Supported CNP Message Subtypes (CMS_)* enum cnp::MSG_TYPE { cnp::MT_INVALID = MAKE_MSG_TYPE(CMT_INVALID, CMS_INVALID), cnp::MT_CONNECT_REQUEST = MAKE_MSG_TYPE(CMT_CONNECT, CMS_REQUEST), cnp::MT_CONNECT_RESPONSE = MAKE_MSG_TYPE(CMT_CONNECT, CMS_RESPONSE), cnp::MT_CREATE_ACCOUNT_REQUEST = MAKE_MSG_TYPE(CMT_CREATE_ACCOUNT, CMS_REQUEST), cnp::MT_CREATE_ACCOUNT_RESPONSE = MAKE_MSG_TYPE(CMT_CREATE_ACCOUNT, CMS_RESPONSE), cnp::MT_LOGON_REQUEST = MAKE_MSG_TYPE(CMT_LOGON, CMS_REQUEST), cnp::MT_LOGON_RESPONSE = MAKE_MSG_TYPE(CMT_LOGON, CMS_RESPONSE), cnp::MT_LOGOFF_REQUEST = MAKE_MSG_TYPE(CMT_LOGOFF,

CMS_REQUEST), cnp::MT_LOGOFF_RESPONSE = MAKE_MSG_TYPE(CMT_LOGOFF, CMS_RESPONSE), cnp::MT_DEPOSIT_REQUEST = MAKE_MSG_TYPE(CMT_DEPOSIT, CMS_REQUEST), cnp::MT_DEPOSIT_RESPONSE = MAKE_MSG_TYPE(CMT_DEPOSIT, CMS_RESPONSE), cnp::MT_WITHDRAWAL_REQUEST = MAKE_MSG_TYPE(CMT_WITHDRAWAL, CMS_REQUEST), cnp::MT_WITHDRAWAL_RESPONSE = MAKE_MSG_TYPE(CMT_WITHDRAWAL, CMS_RESPONSE), cnp::MT_BALANCE_QUERY_REQUEST = MAKE_MSG_TYPE(CMT_BALANCE_QUERY, CMS_REQUEST), cnp::MT_BALANCE_QUERY_RESPONSE = MAKE_MSG_TYPE(CMT_BALANCE_QUERY, CMS_RESPONSE), cnp::MT_TRANSACTION_QUERY_REQUEST = MAKE_MSG_TYPE(CMT_TRANSACTION_QUERY, CMS_REQUEST), cnp::MT_TRANSACTION_QUERY_RESPONSE = MAKE_MSG_TYPE(CMT_TRANSACTION_QUERY, CMS_RESPONSE), cnp::MT_PURCHASE_STAMPS_REQUEST = MAKE_MSG_TYPE(CMT_PURCHASE_STAMPS, CMS_REQUEST), cnp::MT_PURCHASE_STAMPS_RESPONSE = MAKE_MSG_TYPE(CMT_PURCHASE_STAMPS, CMS_RESPONSE) }

- *Constructed Message Type IDs.* enum cnp::CFC_TYPE { cnp::CFC_CONNECT, cnp::CFC_CREDENTIALS, cnp::CFC_FUNCTIONAL, cnp::CFC_ACCOUNT, cnp::CFC_UNDEFINED }
- *Message Facility Code Types (CFC)* enum cnp::CER_TYPE { cnp::CER_SUCCESS = 0, cnp::CER_AUTHENICATION_FAILED = MAKE_ERROR_RESULT(CFC_CONNECT, 0x01), cnp::CER_UNSUPPORTED_PROTOCOL = MAKE_ERROR_RESULT(CFC_CONNECT, 0x02), cnp::CER_INVALID_CLIENT_ID = MAKE_ERROR_RESULT(CFC_CREDENTIALS, 0x01), cnp::CER_INVALID_NAME_PIN = MAKE_ERROR_RESULT(CFC_CREDENTIALS, 0x02), cnp::CER_INVALID_ARGUMENTS = MAKE_ERROR_RESULT(CFC_FUNCTIONAL, 0x01), cnp::CER_CLIENT_NOT_LOGGEDON = MAKE_ERROR_RESULT(CFC_FUNCTIONAL, 0x02), cnp::CER_DRAWER_BLOCKED = MAKE_ERROR_RESULT(CFC_FUNCTIONAL, 0x03), cnp::CER_INSUFFICIENT_FUNDS = MAKE_ERROR_RESULT(CFC_ACCOUNT, 0x01), cnp::CER_ACCOUNT_NOT_FOUND = MAKE_ERROR_RESULT(CFC_ACCOUNT, 0x02), cnp::CER_ACCOUNT_EXISTS = MAKE_ERROR_RESULT(CFC_ACCOUNT, 0x03), cnp::CER_ERROR = (~0) }
- *CNP Error Result Types (CER)* enum cnp::DEPOSIT_TYPE { cnp::DT_INVALID = 0, cnp::DT_CASH = 0x01, cnp::DT_CHECK = 0x02 }
- *CNP Deposit types (DT)* enum cnp::TRANSACTION_TYPE { cnp::TT_INVALID = 0, cnp::TT_DEPOSIT = 0x01, cnp::TT_WITHDRAWAL = 0x02, cnp::TT_STAMP_PURCHASE = 0x03 }

CNP Transaction types (TT) Functions

- bool cnp::Succeeded (cnp::CER_TYPE cerRR) throw ()

Variables

- const WORD cnp::g_wMajorVersion = 1
  Global message sequence number.
- const WORD cnp::g_wMinorVersion = 1
  Protocol minor version (i.e. x.3)
- const DWORD cnp::g_dwValidationKey = 0x00DEAD01
  CNP Validation Key.
- const size_t cnp::MAX_NAME_LEN = 32
  [first,last,email] name field lengths
- const WORD cnp::INVALID_CLIENT_ID = (~0)
  Used for error checking & default initialization.
- const WORD cnp::INVALID_PIN = 0
  Used for error checking & default initialization.

Macro Definition Documentation

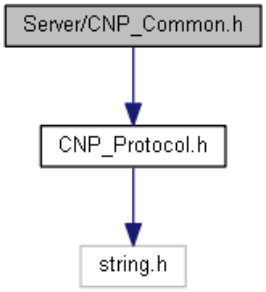
#define COUNTOF(_array) (sizeof(_array) / sizeof(_array[0]))

#define MAKE_MSG_TYPE(type, sub) ((sub << 16) + type)

#define MAKE_ERROR_RESULT(facility, sub) ((facility << 16) + sub)

Server/CNP_Common.h File Reference

Common type definitions.
Include dependency graph for CNP_Common.h:



Functions

- bool IsValidCustomerID (const cnp::QWORD &qwID) throw ()
  Validation helper function.
- bool IsValidPIN (cnp::WORD wPIN) throw ()
  Validation helper function.
- bool IsValidName (const char *szName) throw ()
  Validation helper function.

Variables

- const cnp::QWORD INVALID_CUSTOMER_ID = 0
  for error checking and data initialization
- const cnp::DWORD INVALID_BALANCE = (~0)
  for error checking and data initialization
- const cnp::WORD g_wServerMajorVersion = 1
- const cnp::WORD g_wServerMinorVersion = 1

Detailed Description

Author:
Mark L. Short

Date:
April 10, 2015
April 25, 2015 updated code comments

Function Documentation

bool IsValidCustomerID (const cnp::QWORD & *qwID*) throw)

bool IsValidPIN (cnp::WORD *wPIN*) throw)

bool IsValidName (const char * *szName*) throw)

Variable Documentation

const cnp::QWORD INVALID_CUSTOMER_ID = 0

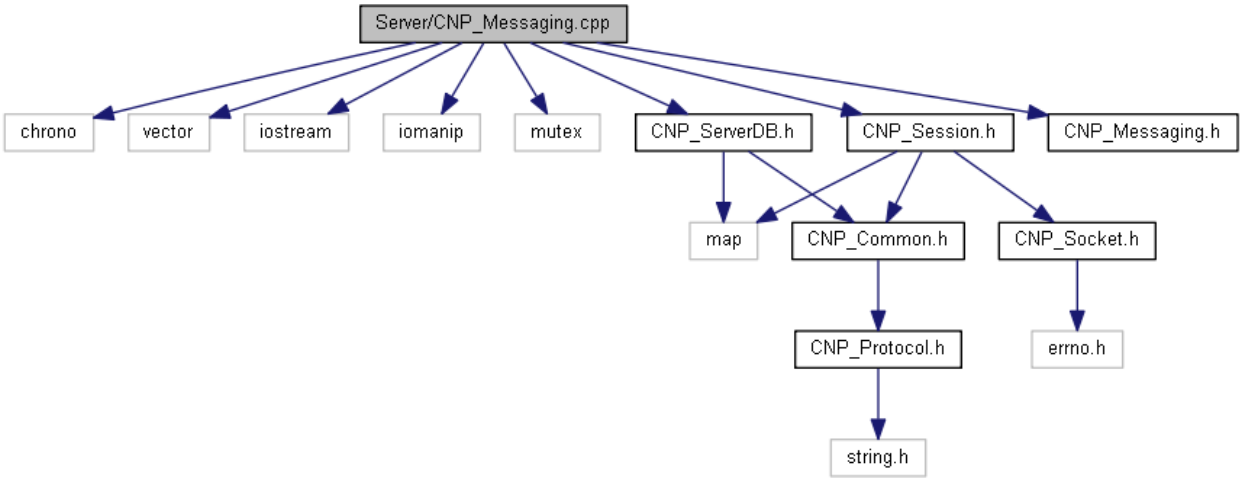
const cnp::DWORD INVALID_BALANCE = (~0)

const cnp::WORD g_wServerMajorVersion = 1

const cnp::WORD g_wServerMinorVersion = 1

Server/CNP_Messaging.cpp File Reference

Server Message processing implementation.
Include dependency graph for CNP_Messaging.cpp:



Functions

- cnp::WORD ProcessConnectRequest (const void *pMsg, size_t cbMsgLen, CNP_Socket *pSocket)
- bool ProcessCreateAccountRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessLogonRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessLogoffRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessDepositRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessWithdrawalRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessBalanceQueryRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessTransactionQueryRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessStampPurchaseRequest (const void *pMsg, size_t cbMsgLen)
- bool ProcessDisconnect (cnp::WORD wClientID)

Variables

- SessionMap_t g_SessionInfo
  Global SessionMap_t instance.
- std::mutex g_SessionMutex
- AccountMap_t g_AccountInfo
- std::mutex g_AccountMutex
- TransactionMap_t g_TransactionInfo
- std::mutex g_TransactionMutex

Detailed Description

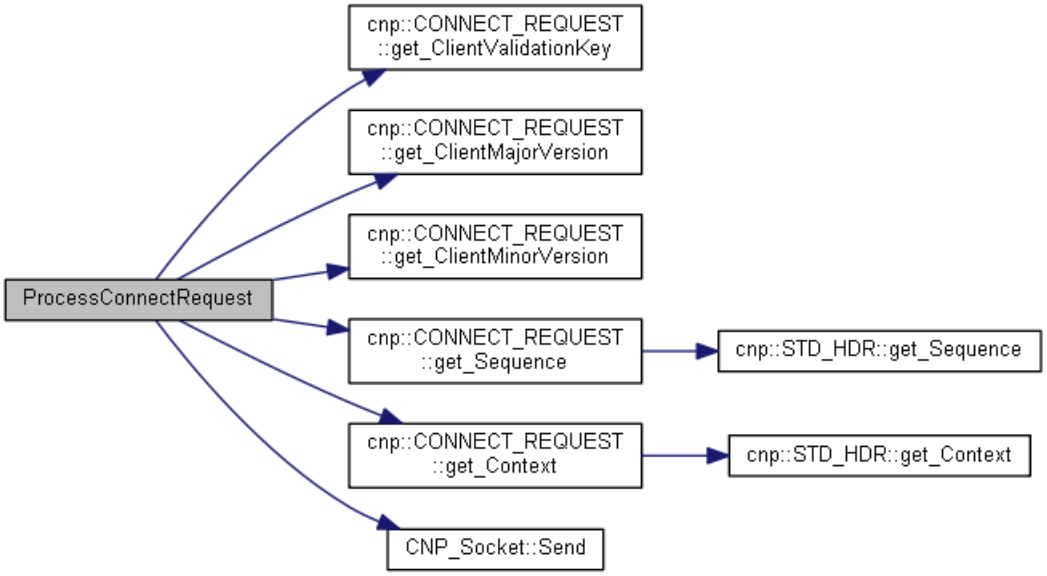
Author:
Mark L. Short

Date:
April 10, 2015

Function Documentation

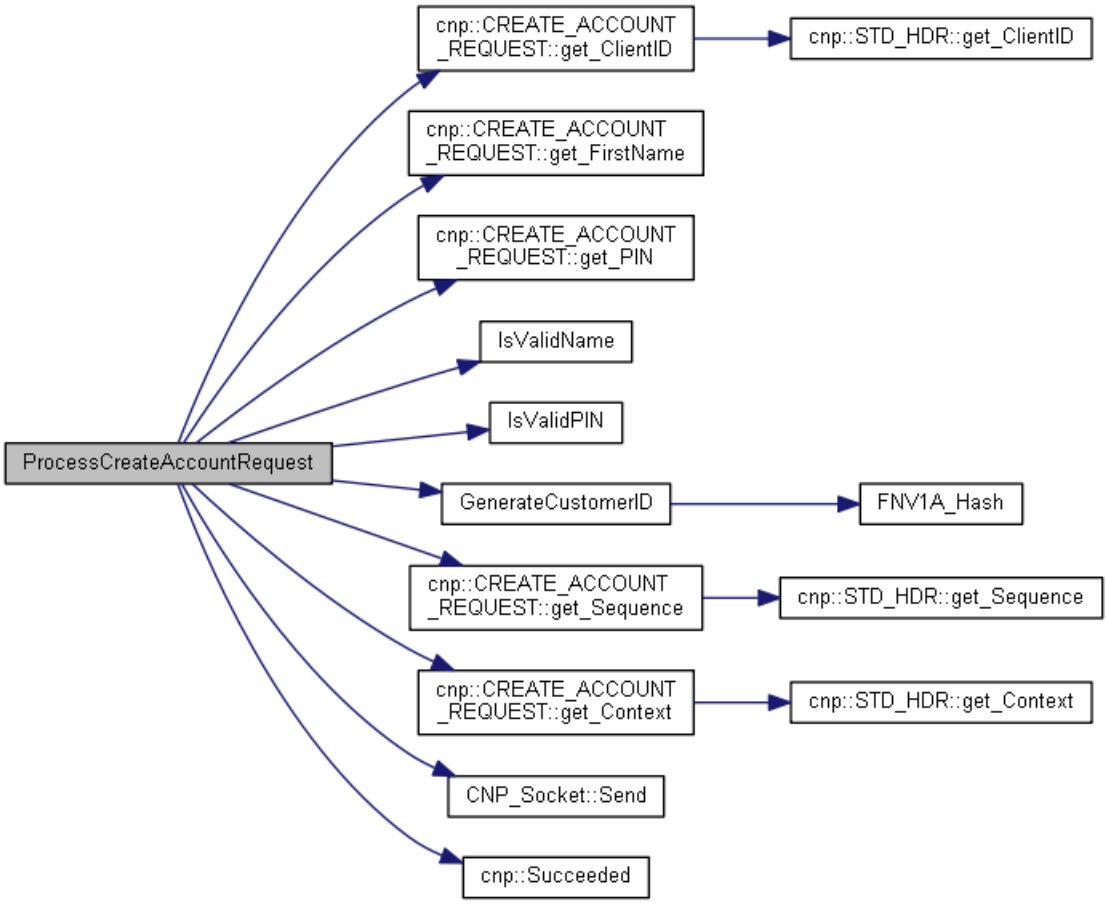
cnp::WORD ProcessConnectRequest (const void * *pMsg*, size_t *cbMsgLen*, CNP_Socket * *pSocket*)

Here is the call graph for this function:



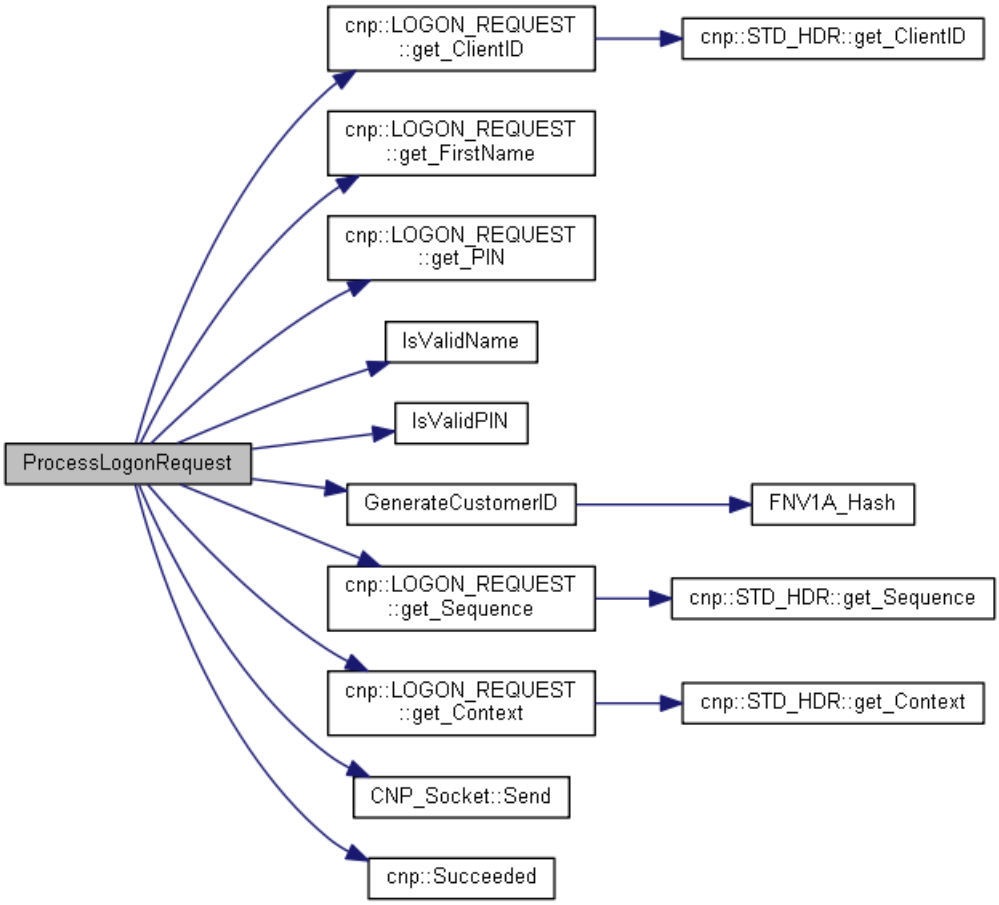
bool ProcessCreateAccountRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



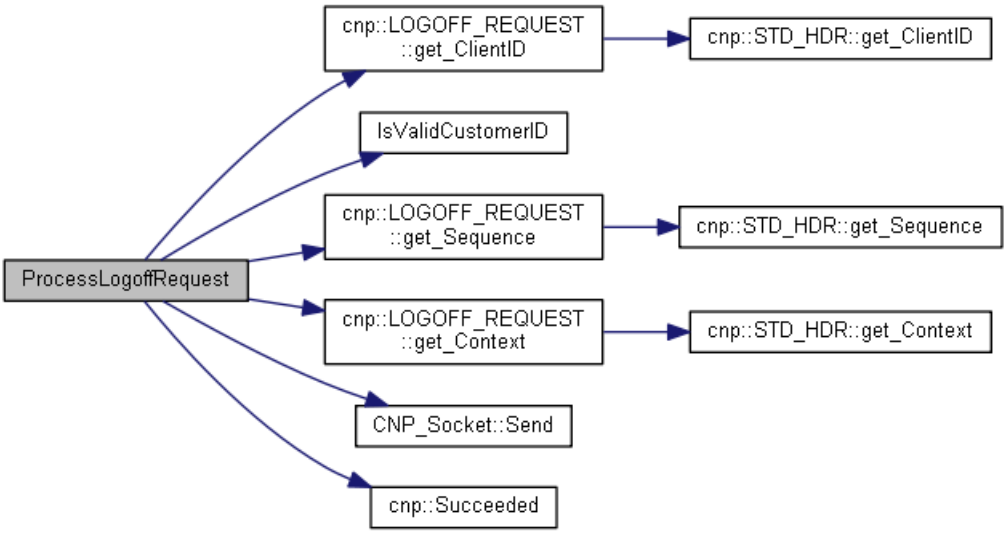
bool ProcessLogonRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



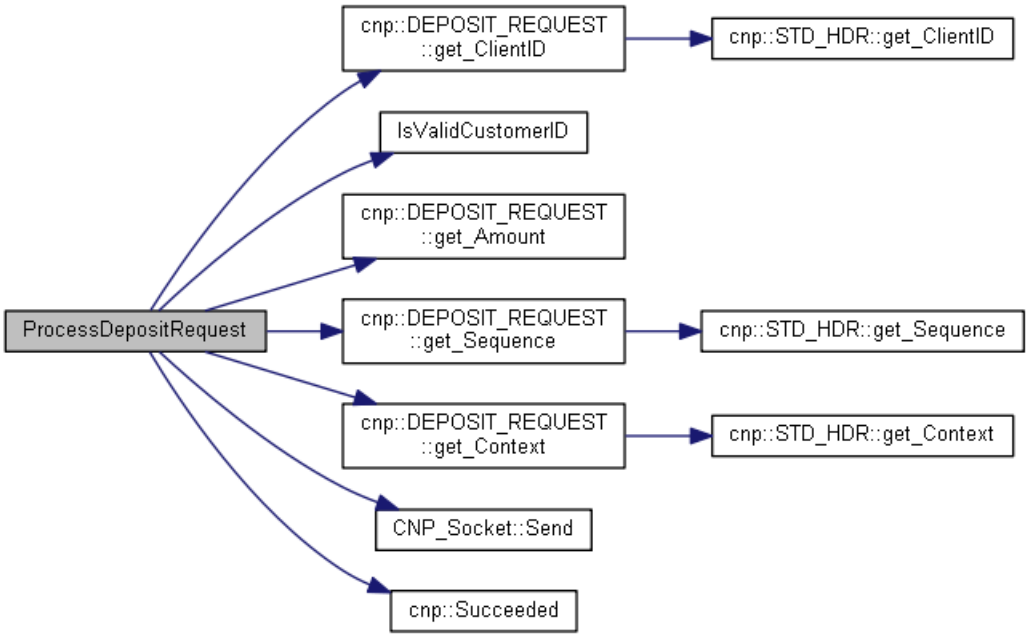
bool ProcessLogoffRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



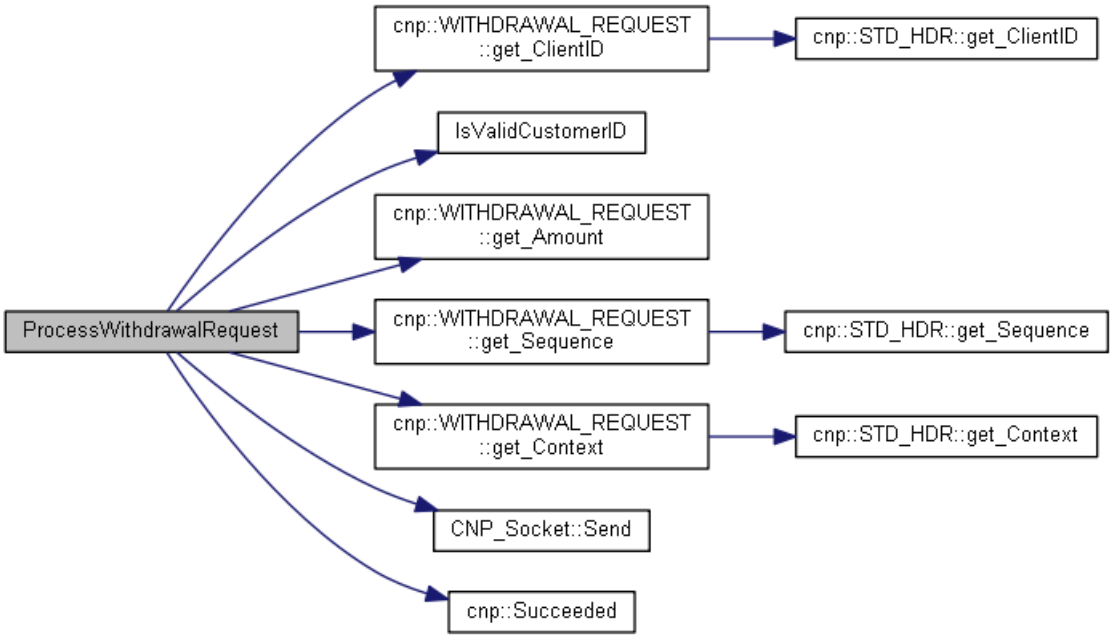
bool ProcessDepositRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



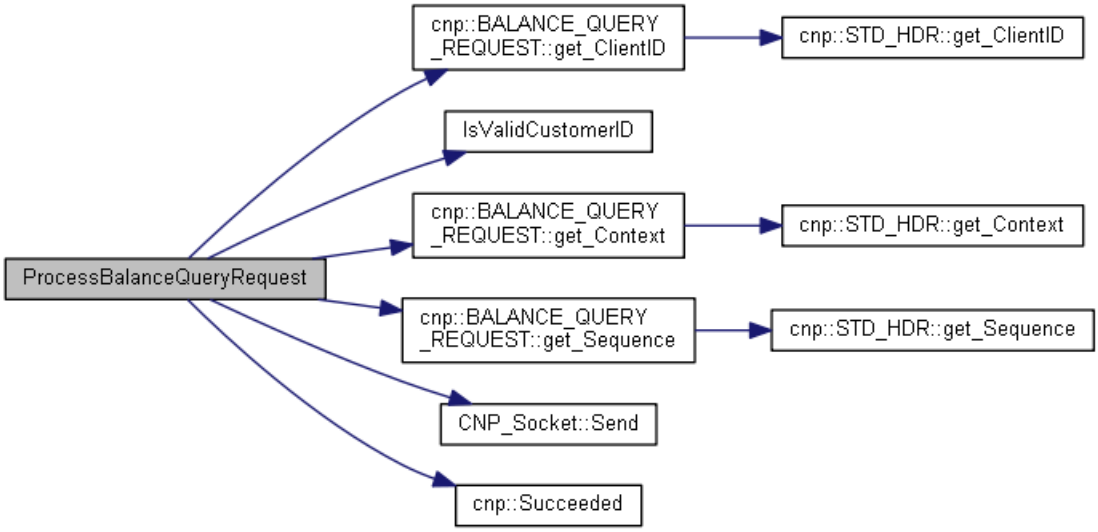
bool ProcessWithdrawalRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



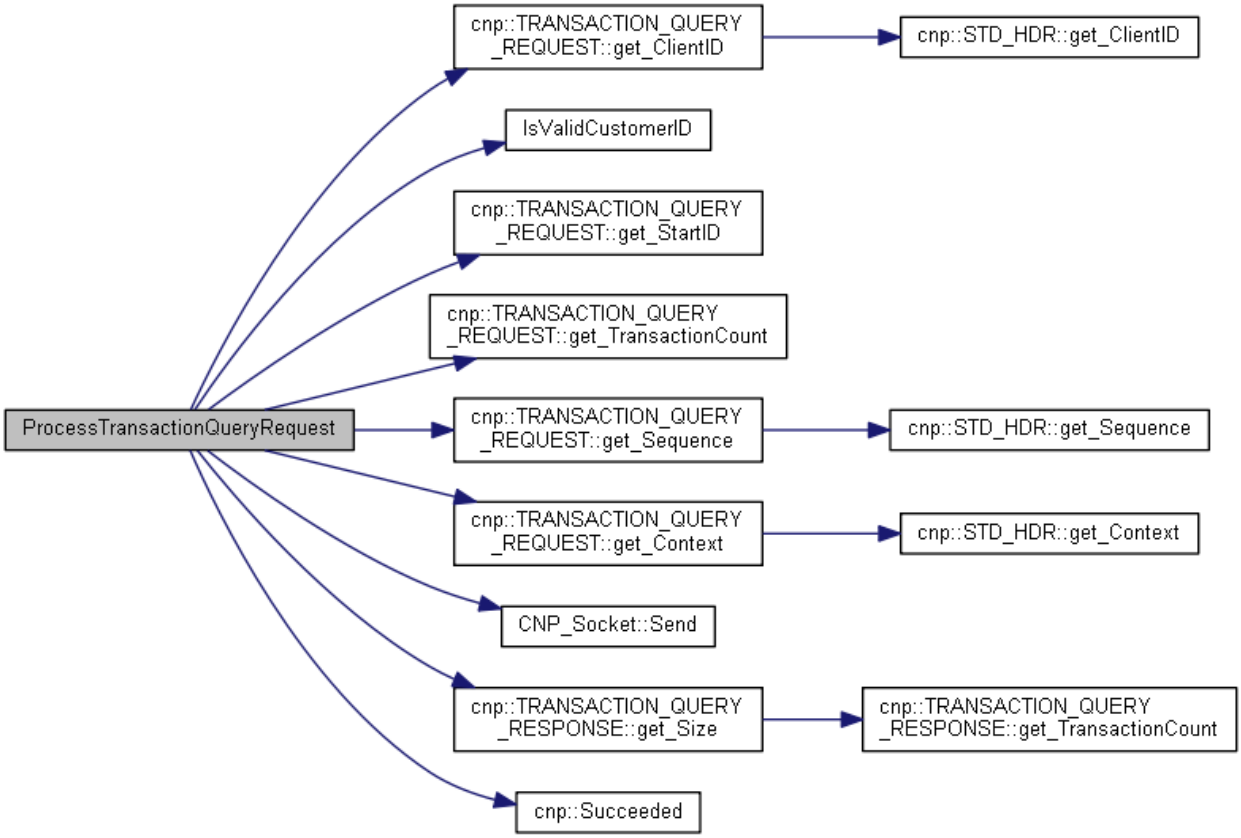
bool ProcessBalanceQueryRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



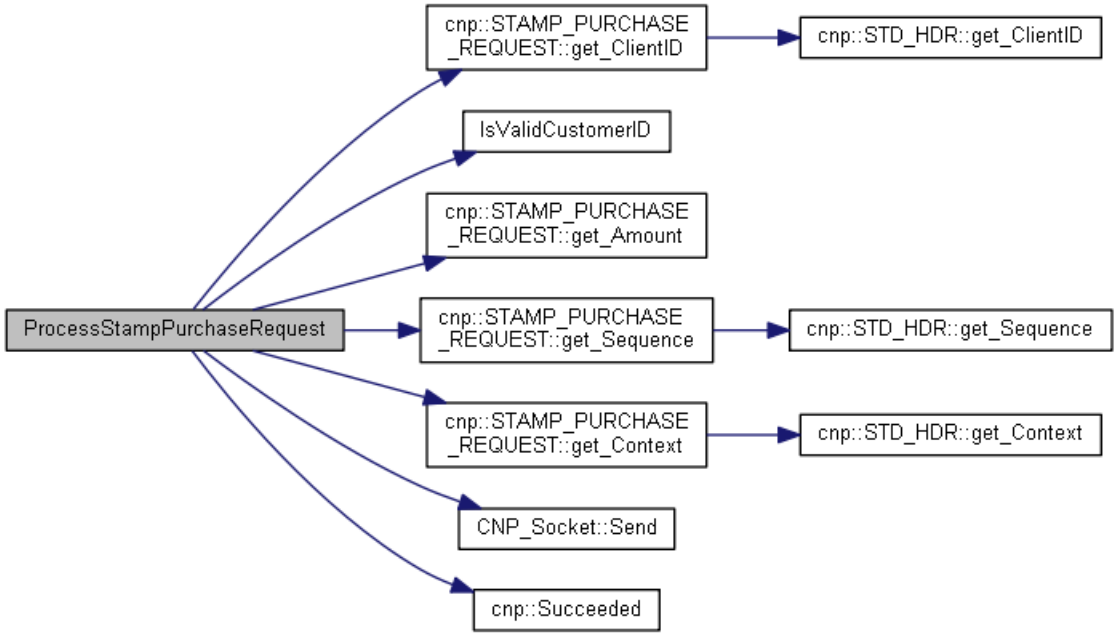
bool ProcessTransactionQueryRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



bool ProcessStampPurchaseRequest (const void * *pMsg*, size_t *cbMsgLen*)

Here is the call graph for this function:



bool ProcessDisconnect (cnp::WORD *wClientID*)

Variable Documentation

SessionMap_t g_SessionInfo

std::mutex g_SessionMutex

AccountMap_t g_AccountInfo

std::mutex g_AccountMutex

TransactionMap_t g_TransactionInfo

std::mutex g_TransactionMutex

Server/CNP_Messaging.h File Reference

Message processing function prototypes.

Functions

- cnp::WORD ProcessConnectRequest (const void *pMsg, size_t cbLen, CNP_Socket *pSocket)
- bool ProcessBalanceQueryRequest (const void *pMsg, size_t cbLen)
- bool ProcessCreateAccountRequest (const void *pMsg, size_t cbLen)
- bool ProcessDepositRequest (const void *pMsg, size_t cbLen)
- bool ProcessLogoffRequest (const void *pMsg, size_t cbLen)
- bool ProcessLogonRequest (const void *pMsg, size_t cbLen)
- bool ProcessStampPurchaseRequest (const void *pMsg, size_t cbLen)
- bool ProcessTransactionQueryRequest (const void *pMsg, size_t cbLen)
- bool ProcessWithdrawalRequest (const void *pMsg, size_t cbLen)
- bool ProcessDisconnect (cnp::WORD wClientID)

Detailed Description

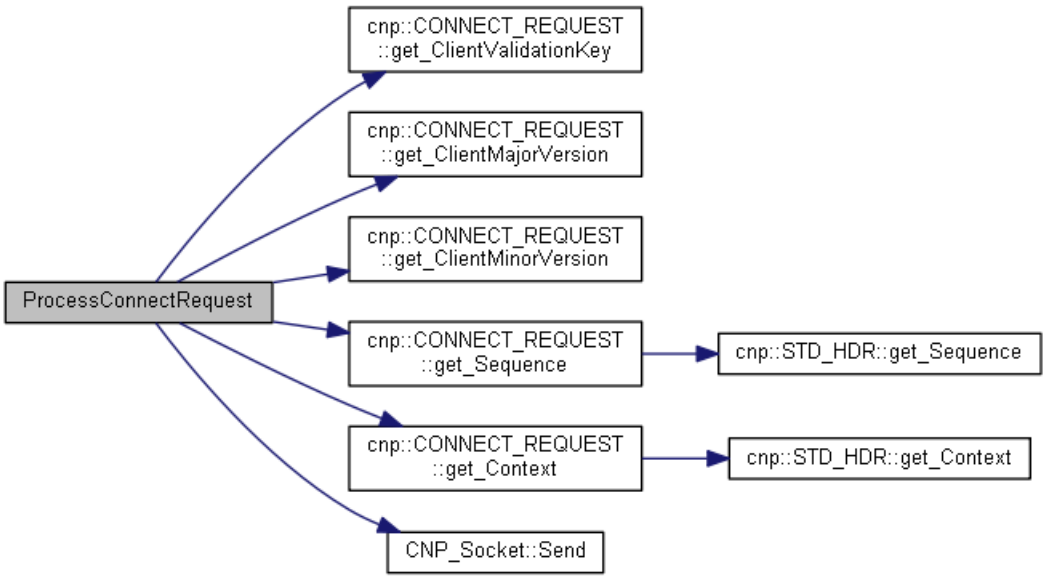
Author:
Mark L. Short

Date:
April 10, 2015

Function Documentation

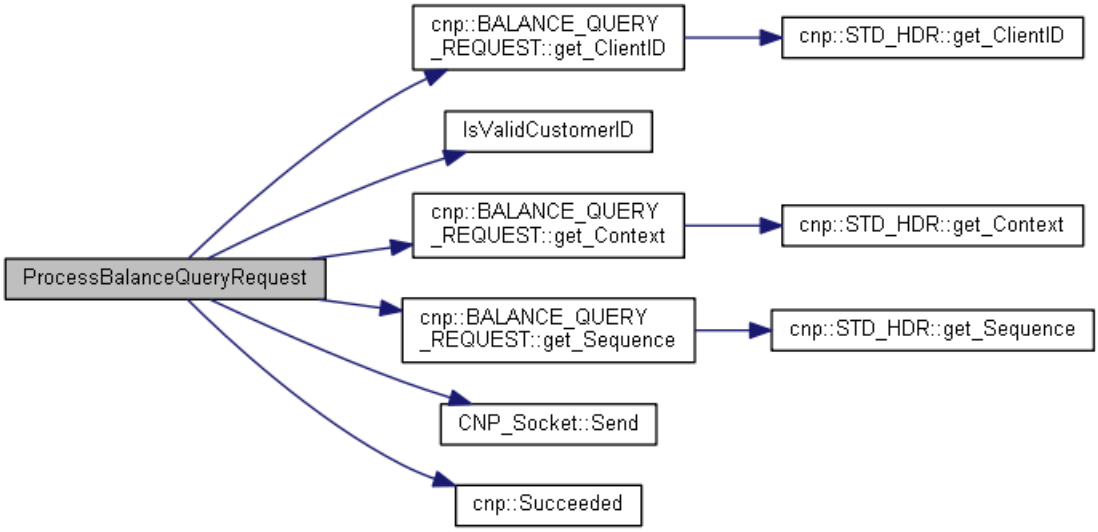
cnp::WORD ProcessConnectRequest (const void * *pMsg*, size_t *cbLen*, CNP_Socket * *pSocket*)

Here is the call graph for this function:



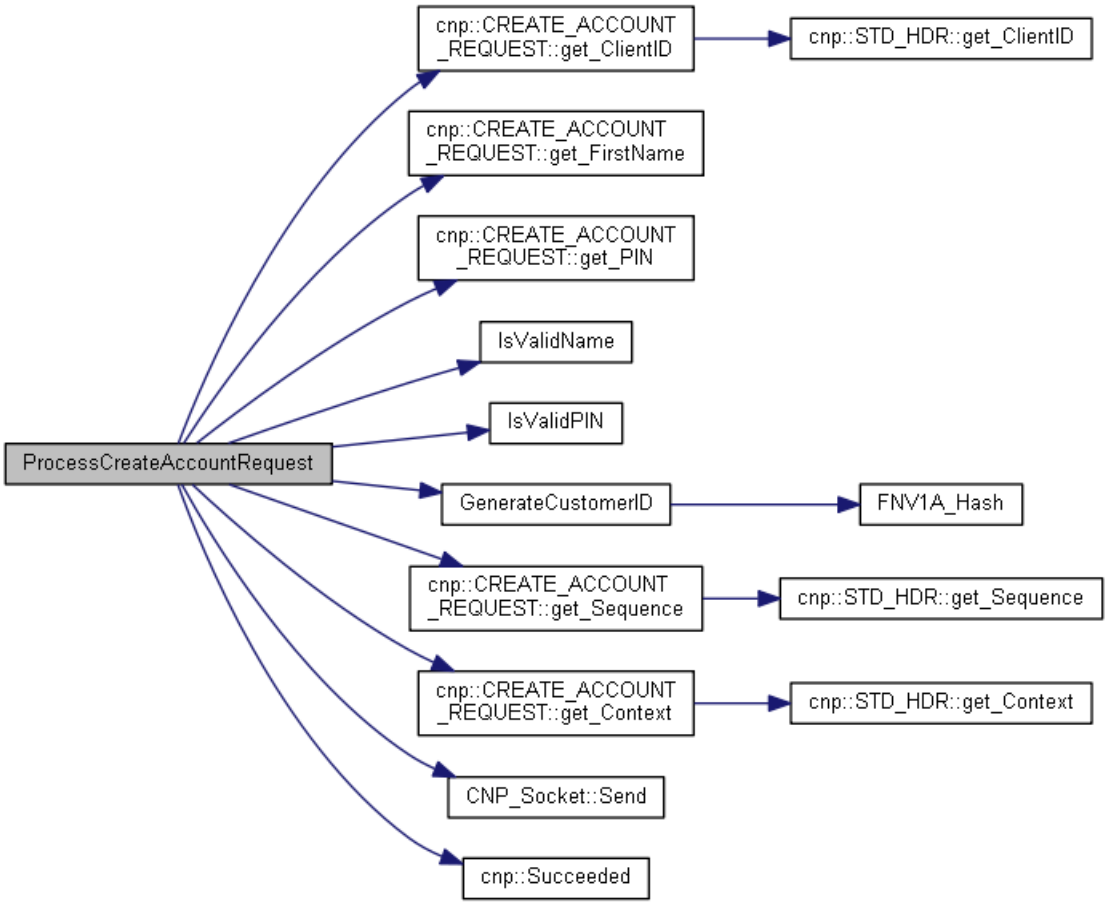
bool ProcessBalanceQueryRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



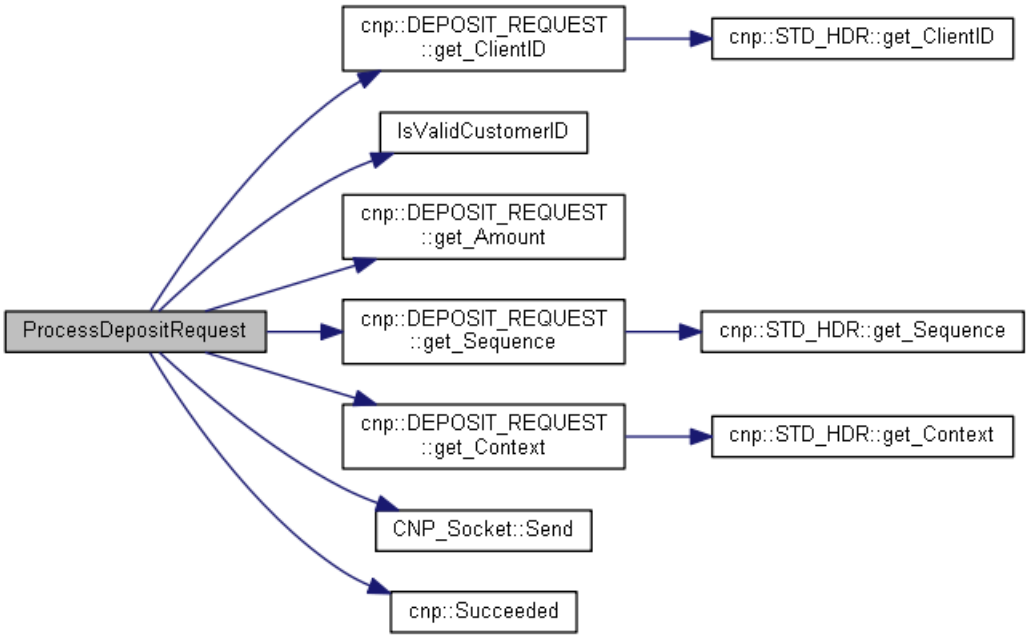
bool ProcessCreateAccountRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



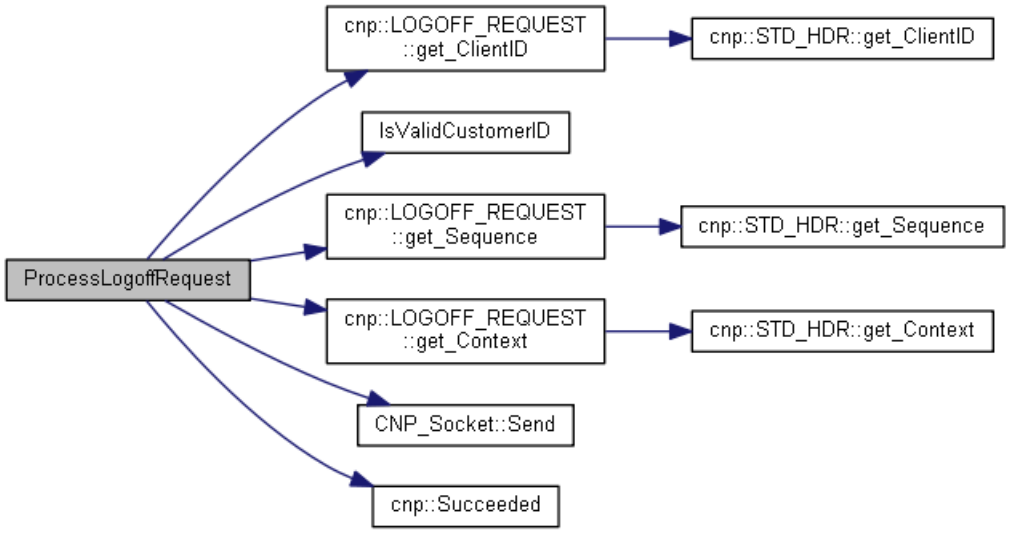
bool ProcessDepositRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



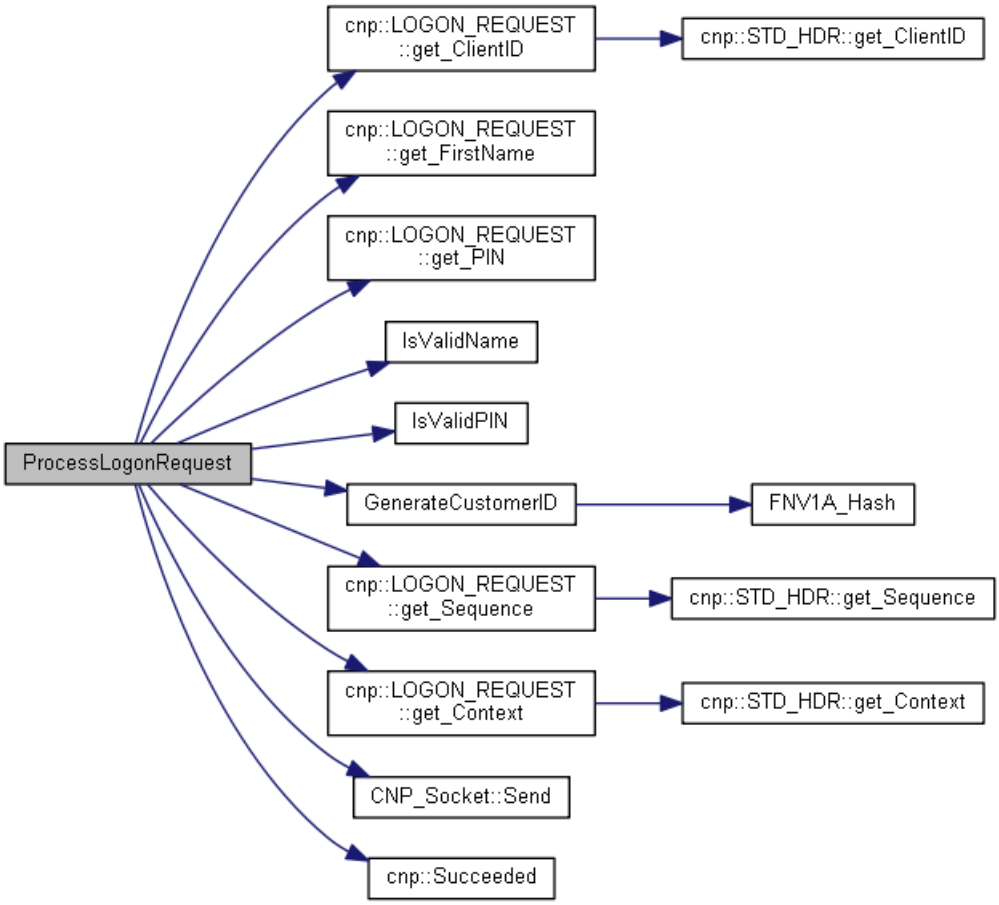
bool ProcessLogoffRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



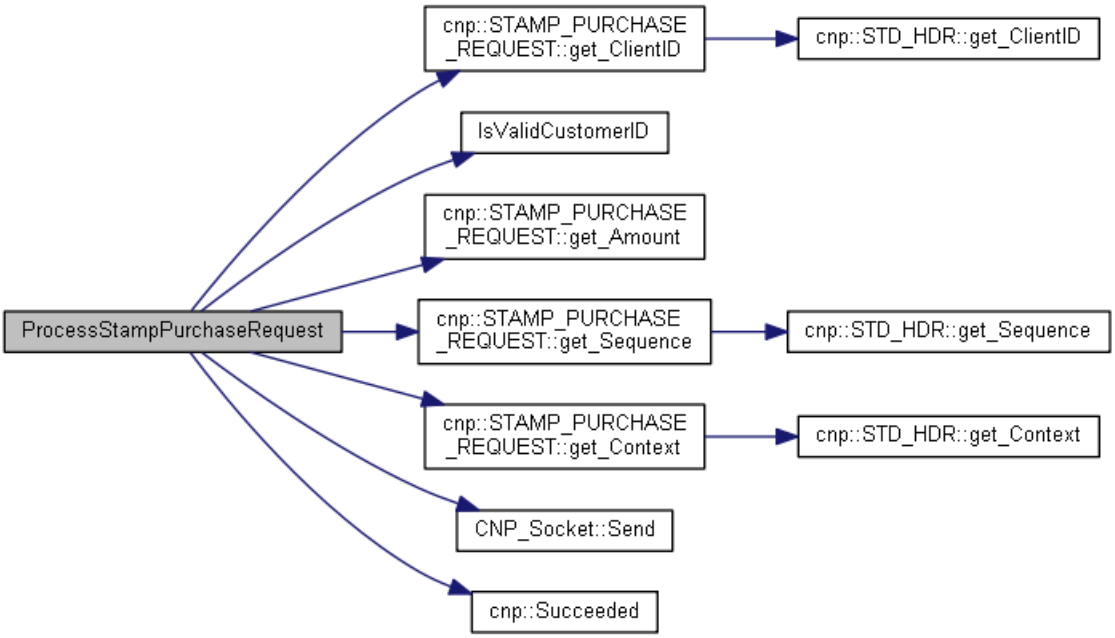
bool ProcessLogonRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



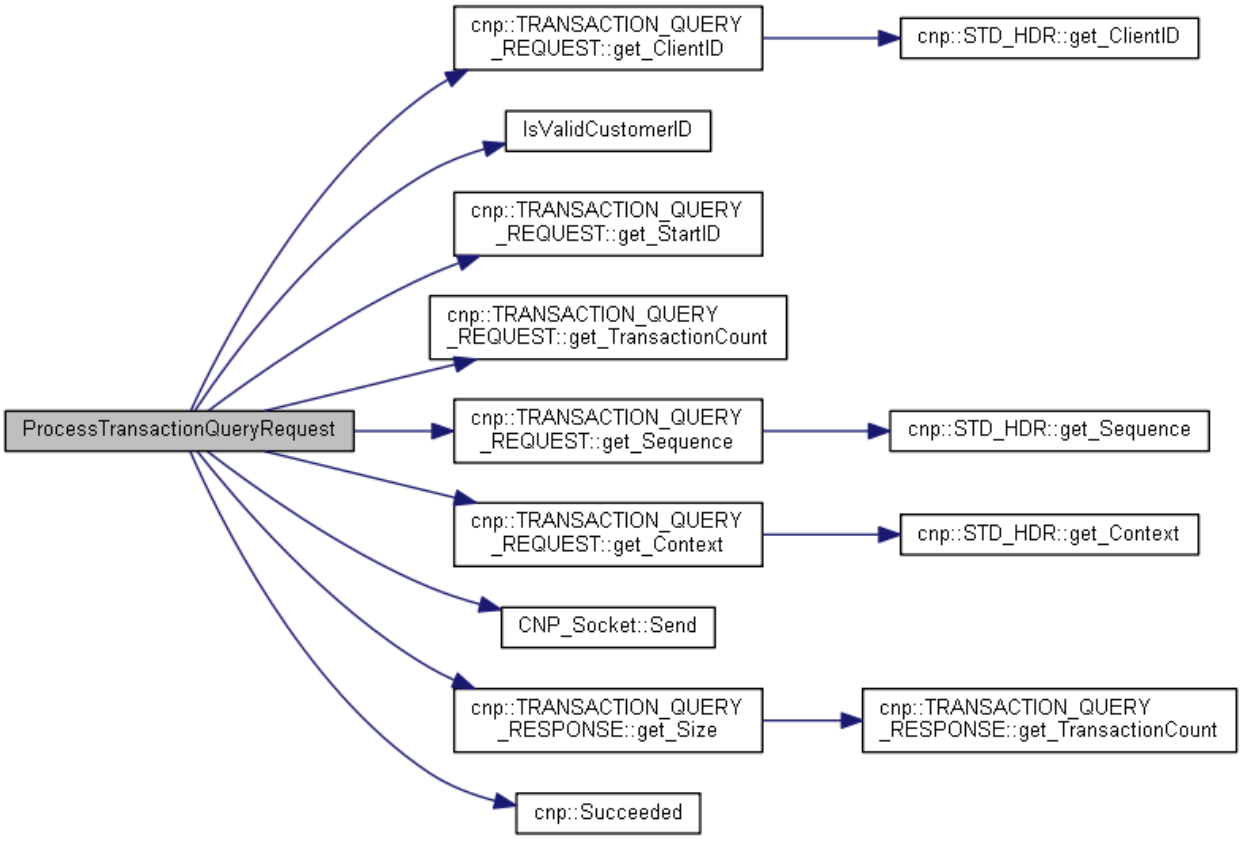
bool ProcessStampPurchaseRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



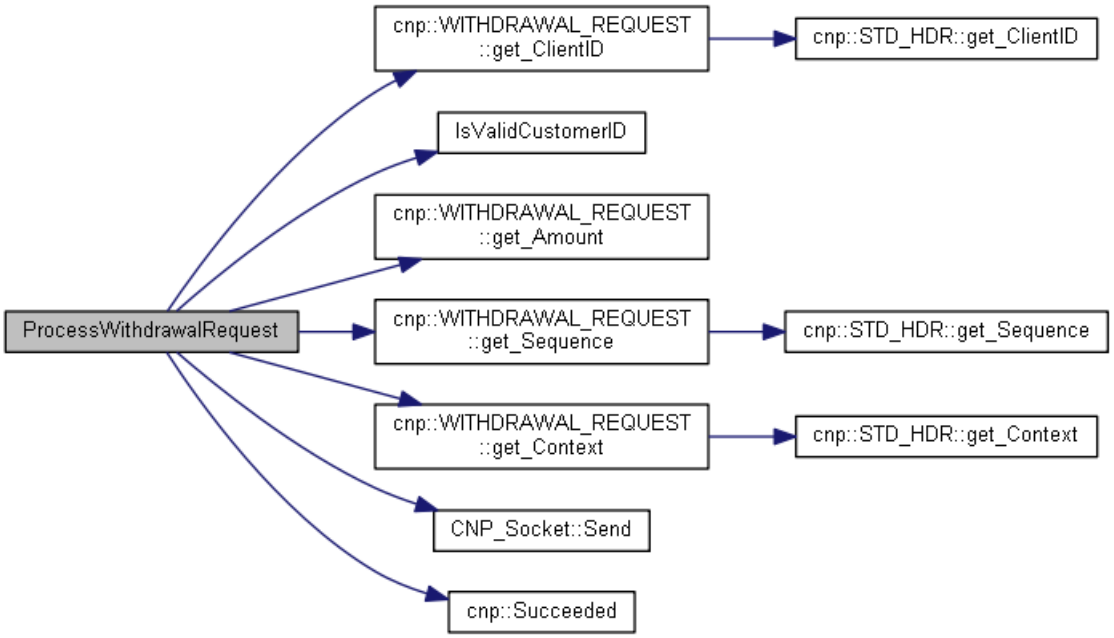
bool ProcessTransactionQueryRequest (const void * *pMsg*, size_t *cbLen*)

Here is the call graph for this function:



bool ProcessWithdrawalRequest (const void * *pMsg*, size_t *cbLen*)

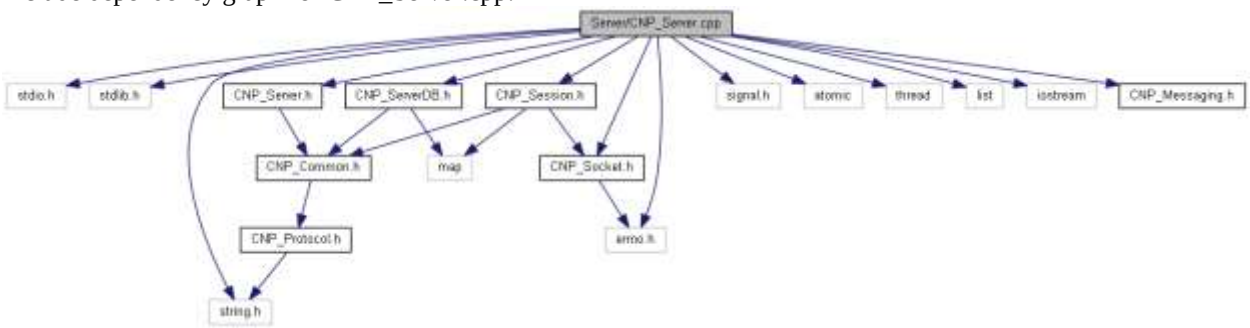
Here is the call graph for this function:



bool ProcessDisconnect (cnp::WORD *wClientID*)

Server/CNP_Server.cpp File Reference

Server Main.
Include dependency graph for CNP_Server.cpp:



Classes

- struct THREAD_INFO

Functions

- void ClientThreadHandler (void *pData)
- void TerminateHandler (int iSignal)
- int main (int argc, char *argv[])

Detailed Description

Author:

Mark L. Short

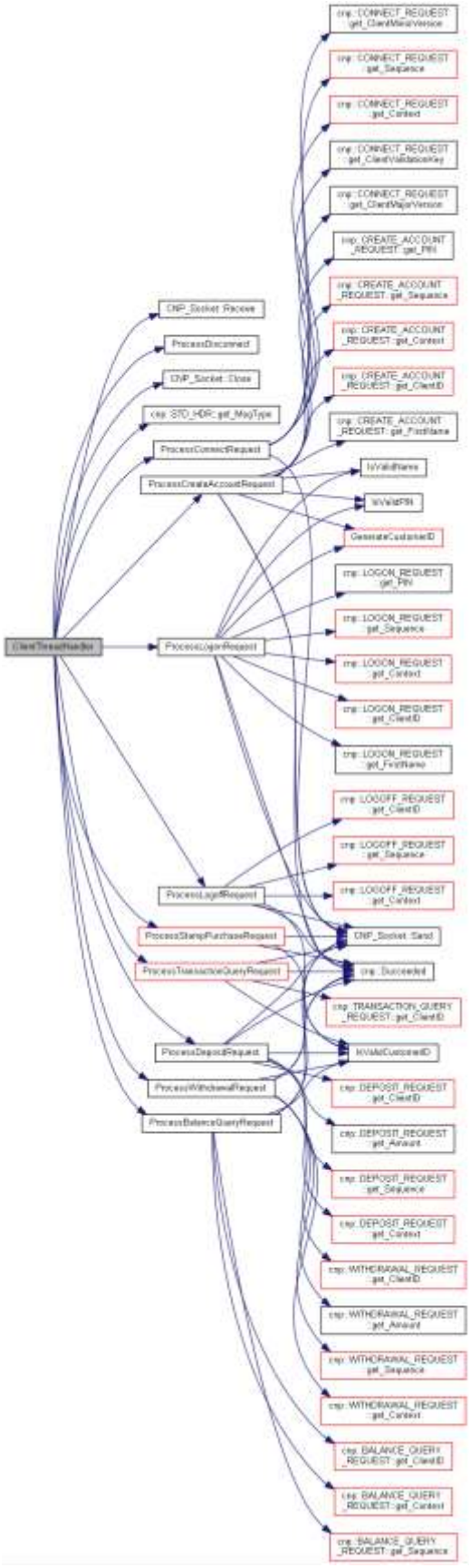
Date:

April 10, 2015

Function Documentation

void ClientThreadHandler (void * *pData*)

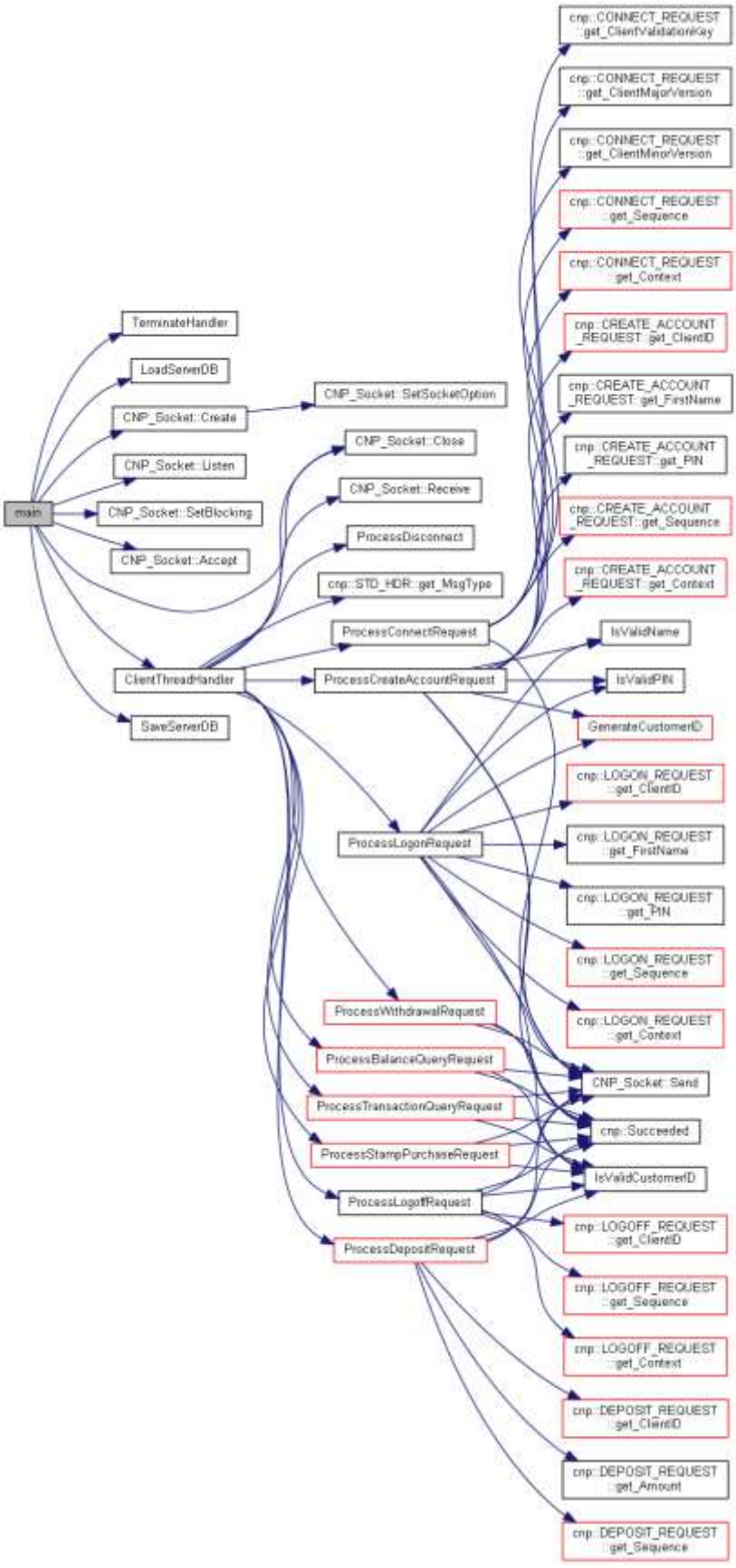
Here is the call graph for this function:


ClientThreadHandler
CNP_Socket::Receive
ProcessDisconnect
CNP_Socket::Close
cnp::STD_HDR::get_MsgType
ProcessConnectRequest
ProcessCreateAccountRequest
ProcessLogonRequest
ProcessLogoffRequest
ProcessStampPurchaseRequest
ProcessTransactionQueryRequest
ProcessDepositRequest
ProcessWithdrawalRequest
ProcessBalanceQueryRequest
cnp::CONNECT_REQUEST::get_ClientMinorVersion
cnp::CONNECT_REQUEST::get_Sequence
cnp::CONNECT_REQUEST::get_Context
cnp::CONNECT_REQUEST::get_ClientValidationKey
cnp::CONNECT_REQUEST::get_ClientMajorVersion
cnp::CREATE_ACCOUNT_REQUEST::get_PIN
cnp::CREATE_ACCOUNT_REQUEST::get_Sequence
cnp::CREATE_ACCOUNT_REQUEST::get_Context
cnp::CREATE_ACCOUNT_REQUEST::get_ClientID
cnp::CREATE_ACCOUNT_REQUEST::get_FirstName
IsValidName
IsValidPIN
GenerateCustomerID
cnp::LOGON_REQUEST::get_PIN
cnp::LOGON_REQUEST::get_Sequence
cnp::LOGON_REQUEST::get_Context
cnp::LOGON_REQUEST::get_ClientID
cnp::LOGON_REQUEST::get_FirstName
cnp::LOGOFF_REQUEST::get_ClientID
cnp::LOGOFF_REQUEST::get_Sequence
cnp::LOGOFF_REQUEST::get_Context
CNP_Socket::Send
cnp::Succeeded
cnp::TRANSACTION_QUERY_REQUEST::get_ClientID
IsValidCustomerID
cnp::DEPOSIT_REQUEST::get_ClientID
cnp::DEPOSIT_REQUEST::get_Amount
cnp::DEPOSIT_REQUEST::get_Sequence
cnp::DEPOSIT_REQUEST::get_Context
cnp::WITHDRAWAL_REQUEST::get_ClientID
cnp::WITHDRAWAL_REQUEST::get_Amount
cnp::WITHDRAWAL_REQUEST::get_Sequence
cnp::WITHDRAWAL_REQUEST::get_Context
cnp::BALANCE_QUERY_REQUEST::get_ClientID
cnp::BALANCE_QUERY_REQUEST::get_Context
cnp::BALANCE_QUERY_REQUEST::get_Sequence

void TerminateHandler (int *iSignal*)

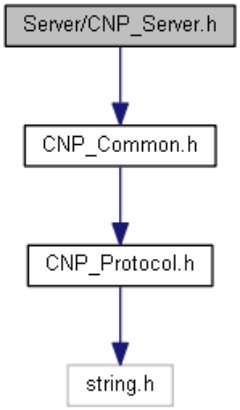
int main (int *argc*, char * *argv*[])

Here is the call graph for this function:


main
TerminateHandler
LoadServerDB
CNP_Socket::Create
CNP_Socket::Listen
CNP_Socket::SetBlocking
CNP_Socket::Accept
SaveServerDB
CNP_Socket::SetSocketOption
CNP_Socket::Close
CNP_Socket::Receive
ProcessDisconnect
cnp::STD_HDR::get_MsgType
ClientThreadHandler
ProcessConnectRequest
ProcessCreateAccountRequest
ProcessLogonRequest
ProcessWithdrawalRequest
ProcessBalanceQueryRequest
ProcessTransactionQueryRequest
ProcessStampPurchaseRequest
ProcessLogoffRequest
ProcessDepositRequest
cnp::CONNECT_REQUEST::get_ClientValidationKey
cnp::CONNECT_REQUEST::get_ClientMajorVersion
cnp::CONNECT_REQUEST::get_ClientMinorVersion
cnp::CONNECT_REQUEST::get_Sequence
cnp::CONNECT_REQUEST::get_Context
cnp::CREATE_ACCOUNT_REQUEST::get_ClientID
cnp::CREATE_ACCOUNT_REQUEST::get_FirstName
cnp::CREATE_ACCOUNT_REQUEST::get_PIN
cnp::CREATE_ACCOUNT_REQUEST::get_Sequence
cnp::CREATE_ACCOUNT_REQUEST::get_Context
IsValidName
IsValidPIN
GenerateCustomerID
cnp::LOGON_REQUEST::get_ClientID
cnp::LOGON_REQUEST::get_FirstName
cnp::LOGON_REQUEST::get_PIN
cnp::LOGON_REQUEST::get_Sequence
cnp::LOGON_REQUEST::get_Context
CNP_Socket::Send
cnp::Succeeded
IsValidCustomerID
cnp::LOGOFF_REQUEST::get_ClientID
cnp::LOGOFF_REQUEST::get_Sequence
cnp::LOGOFF_REQUEST::get_Context
cnp::DEPOSIT_REQUEST::get_ClientID
cnp::DEPOSIT_REQUEST::get_Amount
cnp::DEPOSIT_REQUEST::get_Sequence

Server/CNP_Server.h File Reference

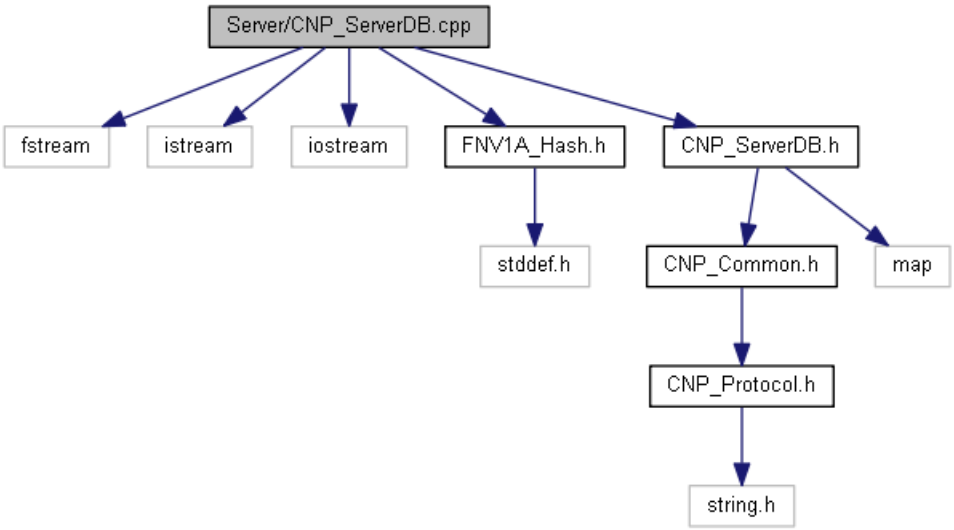
Include dependency graph for CNP_Server.h:



Server/CNP_ServerDB.cpp File Reference

Server DB persistence implementation.

Include dependency graph for CNP_ServerDB.cpp:



Functions

- cnp::QWORD GenerateCustomerID (const char *szFirstName, size_t cbLen, cnp::WORD wPIN)
- template<class _MapType > size_t LoadServerDB (const char *szFileName, _MapType &Container)

Generic template function for loading STL maps.

- template<class _MapType > size_t SaveServerDB (const char *szFileName, const _MapType &Container)
  Generic template function for persisting STL maps.
- size_t LoadServerDB (void)
- size_t SaveServerDB (void)

Variables

- const char g_szAccountDBFileName [] = "..//Data//AccountDB.Dat"
  File name of server ACCOUNT_INFO table store.
- const char g_szTransactDBFileName [] = "..//Data//TransactDB.Dat"
  File name of server TRANSACTION_INFO table store.
- AccountMap_t g_AccountInfo
- TransactionMap_t g_TransactionInfo

Detailed Description

Author:

Mark L. Short

Date:

April 10, 2015
April 25, 2015 updated code comments

Function Documentation

cnp::QWORD GenerateCustomerID (const char * *szFirstName*, size_t *cbLen*, cnp::WORD *wPIN*)

Generate a unique customer ID from a given name + PIN combination

Parameters:

in	*szFirstName*	address of NULL terminated string containing customer's first name
in	*cbLen*	count of bytes (cb) length of first name string
in	*wPIN*	customer's PIN number

Return values:

cnp::QWORD	containing unique 64bit customer ID

Here is the call graph for this function:

template<class _MapType > size_t LoadServerDB (const char * *szFileName*, _MapType & *Container*)

LoadServerDB provides a generic function to use for different map type containers.

Precondition:

_MapType is a std::map<_KeyType, _MappedType> collection
_MappedType implements the get_PrimaryKey() method

Parameters:

in	*szFileName*	address of the NULL terminated string that contains the name of the file to open
in	*Container*	a reference to the std::map container instance to insert loaded file records into

Return values:

size_t	containing the number of records actually loaded

template<class _MapType > size_t SaveServerDB (const char * *szFileName*, const _MapType & *Container*)

SaveServerDB provides a generic function to use for different map type containers.

Precondition:

_MapType is a std::map<_KeyType, _MappedType> collection

Parameters:

in	*szFileName*	address of the NULL terminated string that contains the name of the file to open for saving
in	*Container*	a reference to the std::map container instance to iterate for persistence

Return values:

size_t	containing the number of records actually saved

size_t LoadServerDB (void)

Loads into runtime memory the server database records from the persisted store

Return values:

size_t	containing the number of records loaded

Here is the call graph for this function:

size_t SaveServerDB (void)

Saves the current server database records to persisted store

Return values:

size_t	contain the number of records saved to file

Here is the call graph for this function:

Variable Documentation

const char g_szAccountDBFileName[] = "..//Data//AccountDB.Dat"

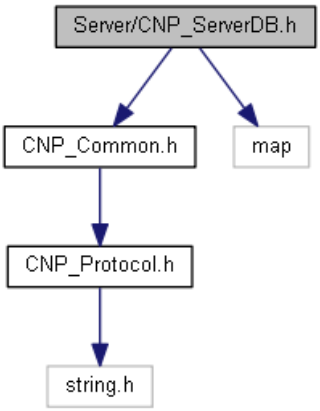
const char g_szTransactDBFileName[] = "..//Data//TransactDB.Dat"

AccountMap_t g_AccountInfo

TransactionMap_t g_TransactionInfo

Server/CNP_ServerDB.h File Reference

ACCOUNT_INFO & TRANSACTION_INFO struct definitions.
Include dependency graph for CNP_ServerDB.h:



Classes

- struct ACCOUNT_INFO
- struct TRANSACTION_INFO

Typedefs

- typedef std::map< ACCOUNT_INFO::key_type, ACCOUNT_INFO > AccountMap_t
- typedef std::map< TRANSACTION_INFO::key_type, TRANSACTION_INFO > TransactionMap_t

Functions

- cnp::QWORD GenerateCustomerID (const char *szFirstName, size_t cbLen, cnp::WORD wPIN)
- size_t LoadServerDB (void)
- size_t SaveServerDB (void)

Detailed Description

Author:

Mark L. Short

Date:

April 10, 2015 original date
April 25, 2015 comments added

Typedef Documentation

typedef std::map<ACCOUNT_INFO::key_type, ACCOUNT_INFO> AccountMap_t

typedef std::map<TRANSACTION_INFO::key_type, TRANSACTION_INFO> TransactionMap_t

---

Function Documentation

cnp::QWORD GenerateCustomerID (const char * *szFirstName*, size_t *cbLen*, cnp::WORD *wPIN*)

Generate a unique customer ID from a given name + PIN combination

Parameters:

in	*szFirstName*	address of NULL terminated string containing customer's first name
in	*cbLen*	count of bytes (cb) length of first name string
in	*wPIN*	customer's PIN number

Return values:

cnp::QWORD	containing unique 64bit customer ID

Here is the call graph for this function:

size_t LoadServerDB (void)

Loads into runtime memory the server database records from the persisted store

Return values:

size_t	containing the number of records loaded

Here is the call graph for this function:

size_t SaveServerDB (void)

Saves the current server database records to persisted store

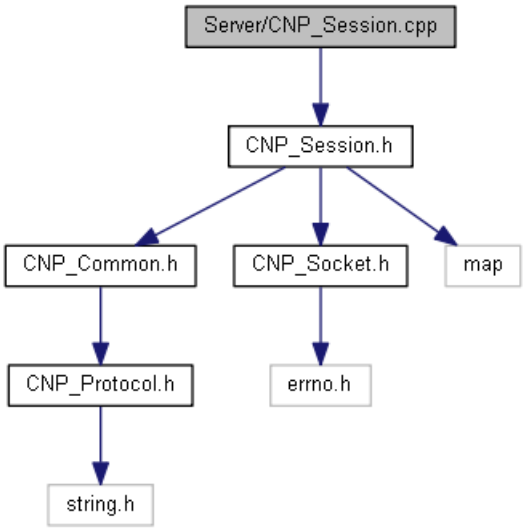
Return values:

size_t	contain the number of records saved to file

Here is the call graph for this function:

Server/CNP_Session.cpp File Reference

SessionMap_t Global Instance.
Include dependency graph for CNP_Session.cpp:



Variables

- SessionMap_t g_SessionInfo
  Global SessionMap_t instance.

Detailed Description

Author:
Mark L. Short

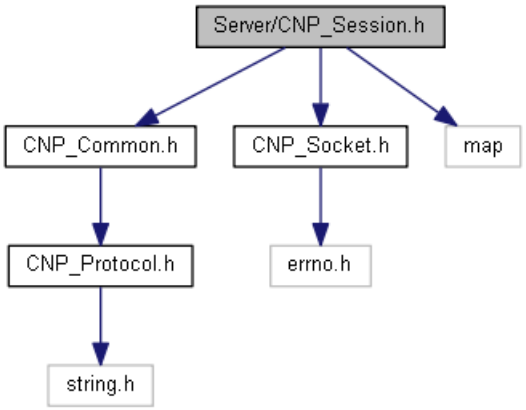
Date:
April 10, 2015

Variable Documentation

SessionMap_t g_SessionInfo

Server/CNP_Session.h File Reference

SESSION_INFO struct definition.
Include dependency graph for CNP_Session.h:



Classes

- struct SESSION_INFO

Typedefs

- typedef std::map< SESSION_INFO::key_type, SESSION_INFO > SessionMap_t

Enumerations

- enum SESSION_STATE { SS_INVALID = 0, SS_CONNECTED, SS_ACCOUNT_CREATED, SS_LOGGED_ON, SS_LOGGED_OFF, SS_DISCONNECTING }

Detailed Description

Author:
Mark L. Short

Date:
April 10, 2015
April 25, 2015 updated comments

Typedef Documentation

typedef std::map<SESSION_INFO::key_type, SESSION_INFO> SessionMap_t

---

Enumeration Type Documentation

enum SESSION_STATE

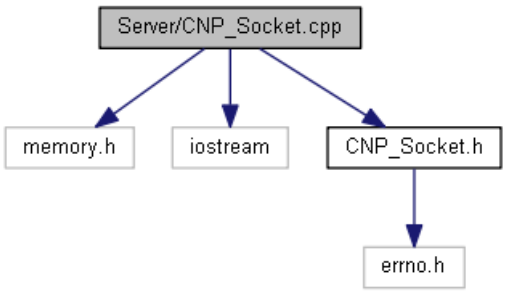
A basic enumeration of allowable session states

Enumerator

SS_INVALID
SS_CONNECTED
SS_ACCOUNT_CREATED
SS_LOGGED_ON
SS_LOGGED_OFF
SS_DISCONNECTING

Server/CNP_Socket.cpp File Reference

Server CNP_Socket class implementation.
Include dependency graph for CNP_Socket.cpp:



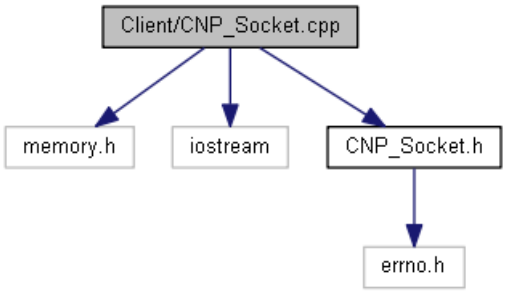
Detailed Description

Author:
Mark L. Short

Date:
April 10, 2015

Client/CNP_Socket.cpp File Reference

Client CNP_Socket class implementation.
Include dependency graph for CNP_Socket.cpp:



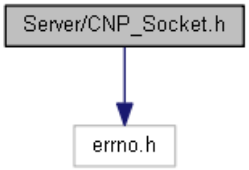
Detailed Description

Author:
Mark L. Short

Date:
April 10, 2015

Server/CNP_Socket.h File Reference

Server CNP_Socket class interface.
Include dependency graph for CNP_Socket.h:



Classes

- class CNP_Socket

Detailed Description

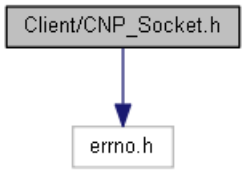
CNP_Socket class provides the basic TCP socket functionality. It supports both Windows & Linux platforms*

Author:
Mark L. Short

Date:
April 10, 2015
April 25, 2015

Client/CNP_Socket.h File Reference

Client CNP_Socket class interface.
Include dependency graph for CNP_Socket.h:



Classes

- class CNP_Socket

Detailed Description

CNP_Socket class provides the basic TCP socket functionality. It supports both Windows & Linux platforms.

Author:

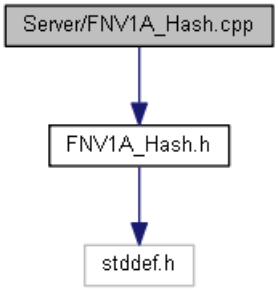
Mark L. Short

Date:

April 10, 2015
April 25, 2015 updated code comments

Server/FNV1A_Hash.cpp File Reference

FNV1A Hash function implementation.
Include dependency graph for FNV1A_Hash.cpp:



Typedefs

- typedef unsigned short WORD
- typedef unsigned long DWORD

Functions

- size_t FNV1A_Hash (const char *pKey, size_t cbLen)

A FNV1a hash implementation.

Detailed Description

Author:

Mark L. Short

Date:

April 10, 2015
April 25, 2015 corrected variable naming that previously implied Key parameter was null-terminated, which it is not required.

See also:

http://en.wikipedia.org/wiki/Fowler%E2%80%93Noll%E2%80%93Vo_hash_function

---

Typedef Documentation

typedef unsigned short WORD

typedef unsigned long DWORD

Function Documentation

size_t FNV1A_Hash (const char * *pKey*, size_t *cbLen*)

Performs a hash computation against a series of bytes. The series may contain embedded NULL characters and does not require a NULL-terminated string as the target.

Parameters:

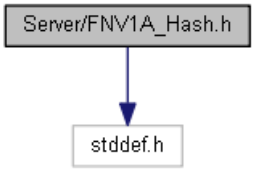
in	*pKey*	address of sequence of bytes
in	*cbLen*	count of byte (cb) length of sequence

Return values:

size_t	containing results of hash

Server/FNV1A_Hash.h File Reference

FNV1A Hash function prototype.
Include dependency graph for FNV1A_Hash.h:



Functions

- size_t FNV1A_Hash (const char *pKey, size_t cbLen)
  A FNV1a hash implementation.

Detailed Description

Author:
Mark L. Short

Date:
April 10, 2015

Cite: The function implementation was based off of the FNV1a hash algorithm, that has been released to public domain.

See also:
http://en.wikipedia.org/wiki/Fowler%E2%80%93Noll%E2%80%93Vo_hash_function

---

Function Documentation

size_t FNV1A_Hash (const char * *pKey*, size_t *cbLen*)

Performs a hash computation against a series of bytes. The series may contain embedded NULL characters and does not require a NULL-terminated string as the target.

Parameters:

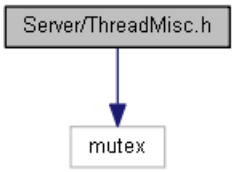
in	*pKey*	address of sequence of bytes
in	*cbLen*	count of byte (cb) length of sequence

Return values:

size_t	containing results of hash

Server/ThreadMisc.h File Reference

Interface for the Thread Related Classes/Templates.
Include dependency graph for ThreadMisc.h:



Classes

- class TLock< _Ty >
- *"Stack-based" Template* class TAutoLock< _Ty >

"Stack-based" Template Typedefs

- typedef TLock< std::recursive_mutex > CLock
- typedef TAutoLock< std::recursive_mutex > CAutoLok
- typedef std::recursive_mutex CAutoCriticalSection

Detailed Description

Author:

Mark L. Short

Date:

February 1, 2006
April 15, 2015 ported to LINUX

Typedef Documentation

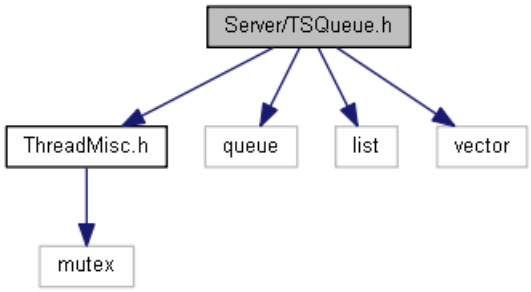
typedef TLock<std::recursive_mutex> CLock

typedef TAutoLock<std::recursive_mutex> CAutoLok

typedef std::recursive_mutex CAutoCriticalSection

Server/TSQueue.h File Reference

interface for the TTSQueue class.
Include dependency graph for TSQueue.h:



Classes

- class TTSQueue< _Ty >

Detailed Description

Thread Safe Queue with Associated Critical Section.

Author:
Mark L. Short

Date:
February 1, 2006

Index