opentxs

Generated by Doxygen 1.9.3

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 opentxs Namespace Reference

NOLINTBEGIN(modernize-concat-nested-namespaces)

Namespaces

- namespace ui

5.1.1 Detailed Description

NOLINTBEGIN(modernize-concat-nested-namespaces)

5.2 opentxs::ui Namespace Reference

Classes

- class AccountActivity
- class AccountCurrency
- class AccountList
- class AccountListItem
- class AccountSummary
- class AccountSummaryItem
- class AccountTree
- class AccountTreeItem
- class ActivitySummary
- class ActivitySummaryItem
- class ActivityThread
- class ActivityThreadItem
- class BalanceItem
- class BlockchainAccountStatus
- class BlockchainSelection
- class BlockchainSelectionItem

- class BlockchainStatistics
- class BlockchainStatisticsItem
- class BlockchainSubaccount
- class BlockchainSubaccountSource
- class BlockchainSubchain
- class Contact
- class ContactItem
- class ContactList
- class ContactListItem
- class ContactSection
- class ContactSubsection
- class IssuerItem
- class List
- class ListRow
- class MessagableList
- class NymList
- class NymListItem
- class PayableList
- class PayableListItem
- class Profile
- class ProfileItem
- class ProfileSection
- class ProfileSubsection
- class SeedTree
- class SeedTreeItem
- class SeedTreeNym
- class UnitList
- class UnitListItem
- class Widget

Enumerations

- enum class **Blockchains** : std::uint8_t { **All** = 0 , **Main** = 1 , **Test** = 2 }

5.2.1 Detailed Description

This class is not yet in use. (Coming soon). The purpose is to provide a tree model for the UI to display the wallet's accounts in a tree, as a replacement for the simple list UI currently in use (AccountList). Each row in the AccountTree is an AccountCurrency, which contains all of the accounts of a given unit type. Each row in the AccountCurrency model contains an AccountTreeItem representing each of the accounts in the wallet of that currency type. For example, AccountTree may contain an AccountCurrency row for Bitcoin and an AccountCurrency row for Ethereum. The row for Bitcoin contains a list of Bitcoin accounts, and the row for Ethereum contains a list of Ethereum accounts (each represented as an AccountTreeItem).

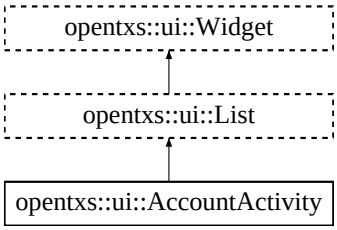
Chapter 6

Class Documentation

6.1 opentxs::ui::AccountActivity Class Reference

`#include <AccountActivity.hpp>`

Inheritance diagram for opentxs::ui::AccountActivity:



Public Types

- using **Scale** = unsigned int

Public Member Functions

- virtual auto **AccountID** () const noexcept -> UnallocatedCString=0

  returns the account ID.
- virtual auto **Balance** () const noexcept -> const Amount=0

  returns the account's balance as an Amount object.
- virtual auto **BalancePolarity** () const noexcept -> int=0

  returns Polarity, since the account balance can be positive or negative.
- virtual auto **ContractID** () const noexcept -> UnallocatedCString=0

  returns For off-chain assets, returns the unit definition ID for the associated asset contract.
- virtual auto **DepositChains** () const noexcept -> UnallocatedVector< blockchain::Type >=0
- virtual auto **DisplayBalance** () const noexcept -> UnallocatedCString=0

  returns a string containing the account balance formatted for display in the UI.
- virtual auto **DisplayUnit** () const noexcept -> UnallocatedCString=0

  returns a string containing the account's unit type formatted for display in the UI.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::BalanceItem >=0

- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  returns Display name for the account.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::BalanceItem >=0
- virtual auto **NotaryID** () const noexcept -> UnallocatedCString=0

  For off-chain assets, this returns the off-chain account's Notary ID.
- virtual auto **NotaryName** () const noexcept -> UnallocatedCString=0

  For off-chain assets, this returns the off-chain account's display name.
- virtual auto **SyncPercentage** () const noexcept -> double=0

  returns Account's current synchronization progress.
- virtual auto **SyncProgress** () const noexcept -> std::pair< int, int >=0

  returns Account's ot::blockchain::Type and related current synchronization progress.
- virtual auto **Type** () const noexcept -> AccountType=0

  Off-chain accounts are type 1, issuer accounts are type 2, and blockchain accounts are type 3.
- virtual auto **Unit** () const noexcept -> UnitType=0

  returns UnitType for the account.
- virtual auto **ValidateAddress** (const UnallocatedCString &text) const noexcept -> bool=0

  returns true if the supplied address text is valid recipient for sends originating from this account.
- virtual auto **ValidateAmount** (const UnallocatedCString &text) const noexcept -> UnallocatedCString=0

  Input is an amount string from user input. Output is the re-formatted amount string.

DepositAddress

Returns

A new receiving address.

- virtual auto **DepositAddress** () const noexcept -> UnallocatedCString=0
- virtual auto **DepositAddress** (const blockchain::Type chain) const noexcept -> UnallocatedCString=0

Send

Send funds from the account to an address or contact.

Parameters

address	*Recipient address as string.*
amount	*Amount as object or string.*
contact	*Identifier containing recipient's Contact ID.*
memo	*Optional memo field as string.*

Returns

Whether or not the send was successful.

- virtual auto **Send** (const Identifier &contact, const Amount &amount, const UnallocatedCString &memo={}) const noexcept -> bool=0
- virtual auto **Send** (const Identifier &contact, const UnallocatedCString &amount, const UnallocatedCString &memo={}, Scale scale=0) const noexcept -> bool=0
- virtual auto **Send** (const UnallocatedCString &address, const Amount &amount, const UnallocatedCString &memo={}) const noexcept -> bool=0
- virtual auto **Send** (const UnallocatedCString &address, const UnallocatedCString &amount, const UnallocatedCString &memo={}, Scale scale=0) const noexcept -> bool=0

6.1.1 Detailed Description

This model manages a set of rows containing Balance Items for the account. It is also an easy way to access account-level data such as balance or receiving address.

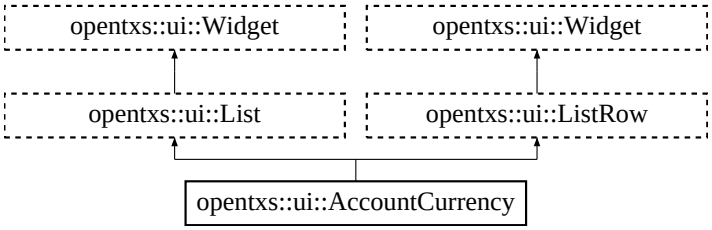
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountActivity.hpp

6.2 opentxs::ui::AccountCurrency Class Reference

`#include <AccountCurrency.hpp>`

Inheritance diagram for opentxs::ui::AccountCurrency:



Public Member Functions

- virtual auto **Currency** () const noexcept -> UnitType=0
- virtual auto **Debug** () const noexcept -> UnallocatedCString=0
- virtual auto **First** () const noexcept -> SharedPimpl< AccountTreeItem >=0
- virtual auto **Name** () const noexcept -> UnallocatedCString=0
- virtual auto **Next** () const noexcept -> SharedPimpl< AccountTreeItem >=0

6.2.1 Detailed Description

The AccountTree model contains a list of AccountCurrency models. AccountCurrency is for iterating and displaying the accounts available in the wallet for a given unit type. For example, all of the Bitcoin accounts. Or all of the Ethereum accounts. Etc. Each row in the AccountCurrency model contains an AccountTreeItem, representing an individual account.

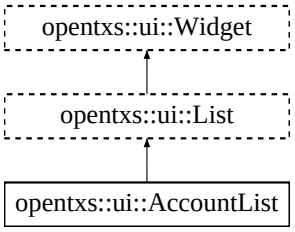
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountCurrency.hpp

6.3 opentxs::ui::AccountList Class Reference

`#include <AccountList.hpp>`

Inheritance diagram for opentxs::ui::AccountList:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::AccountListItem >=0
  returns the first row, containing a valid AccountListItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::AccountListItem >=0
  returns the next row, containing a valid AccountListItem or an empty smart pointer (if at end of list).

6.3.1 Detailed Description

This model manages a set of rows containing a list of accounts for the UI. Each row contains an AccountListItem.

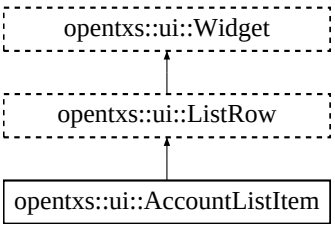
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountList.hpp

6.4 opentxs::ui::AccountListItem Class Reference

`#include <AccountListItem.hpp>`

Inheritance diagram for opentxs::ui::AccountListItem:



Public Member Functions

- virtual auto **AccountID** () const noexcept -> UnallocatedCString=0

  Returns the AccountID for the account.
- virtual auto **Balance** () const noexcept -> Amount=0

  Returns the balance of the account as an Amount object.
- virtual auto **ContractID** () const noexcept -> UnallocatedCString=0

  Returns the ContractID when relevant.
- virtual auto **DisplayBalance** () const noexcept -> UnallocatedCString=0

  Returns the balance of the account as a formatted string for display in the UI.
- virtual auto **DisplayUnit** () const noexcept -> UnallocatedCString=0

  Returns the unit type of the account ("bitcoin" etc) as a formatted string for display in the UI.
- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns display name for the account.
- virtual auto **NotaryID** () const noexcept -> UnallocatedCString=0

  Returns the NotaryID for the account when relevant. (For off-chain accounts).
- virtual auto **NotaryName** () const noexcept -> UnallocatedCString=0

  Returns the display name for the account's notary, when relevant. (For off-chain accounts).
- virtual auto **Type** () const noexcept -> AccountType=0

  Returns the account type. (Issuer account, off-chain account, or blockchain account).
- virtual auto **Unit** () const noexcept -> UnitType=0

  Returns the unit type of the account as an enum.

6.4.1 Detailed Description

This model manages an AccountListItem, representing a single row in the wallet UI's list of accounts.

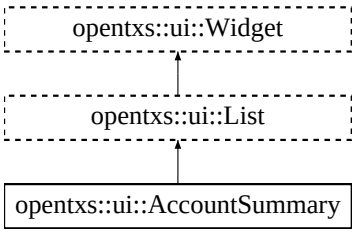
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountListItem.hpp

6.5 opentxs::ui::AccountSummary Class Reference

`#include <AccountSummary.hpp>`

Inheritance diagram for opentxs::ui::AccountSummary:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::IssuerItem >=0
  returns the first row, containing a valid IssuerItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::IssuerItem >=0
  returns the next row, containing a valid IssuerItem or an empty smart pointer (if at end of list).

6.5.1 Detailed Description

This model manages the AccountSummary, containing connection and trusted status for various issuers. NOTE: this model is being updated to manage a list of AccountSummaryItem instead of IssuerItem.

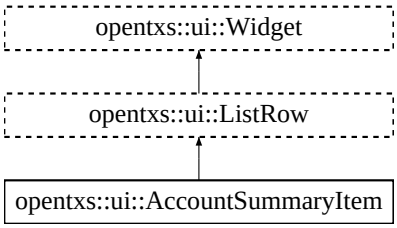
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountSummary.hpp

6.6 opentxs::ui::AccountSummaryItem Class Reference

`#include <AccountSummaryItem.hpp>`

Inheritance diagram for opentxs::ui::AccountSummaryItem:



Public Member Functions

- virtual auto **AccountID** () const noexcept -> UnallocatedCString=0
  Returns the AccountID as a string.
- virtual auto **Balance** () const noexcept -> Amount=0
  Returns the account's balance as an Amount object.
- virtual auto **DisplayBalance** () const noexcept -> UnallocatedCString=0
  Returns the balance of the account, formatted as a string for display in the UI.
- virtual auto **Name** () const noexcept -> UnallocatedCString=0
  Returns the display name of the account.

6.6.1 Detailed Description

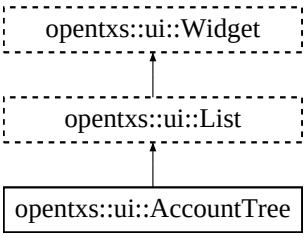
This model manages the AccountSummaryItem, which is a single row on the AccountSummary model and which represents a single account in the wallet.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountSummaryItem.hpp

6.7 opentxs::ui::AccountTree Class Reference

Inheritance diagram for opentxs::ui::AccountTree:



Public Member Functions

- virtual auto **Debug** () const noexcept -> UnallocatedCString=0

  returns debug information relevant to the Currency of the current row.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< AccountCurrency >=0

  returns the first row, containing a valid AccountCurrency or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< AccountCurrency >=0

  returns the next row, containing a valid AccountCurrency or an empty smart pointer (if at end of list).
- virtual auto **Owner** () const noexcept -> const identifier::Nym &=0

  returns the NymID of the wallet owner as an identifier::Nym object.

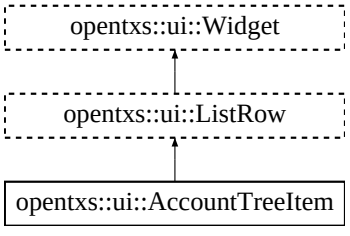
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountTree.hpp

6.8 opentxs::ui::AccountTreeItem Class Reference

`#include <AccountTreeItem.hpp>`

Inheritance diagram for opentxs::ui::AccountTreeItem:



Public Member Functions

- virtual auto **AccountID** () const noexcept -> UnallocatedCString=0

  Returns the AccountID for the account.
- virtual auto **Balance** () const noexcept -> Amount=0

  Returns the balance of the account as an Amount object.
- virtual auto **ContractID** () const noexcept -> UnallocatedCString=0

  Returns the ContractID when relevant.
- virtual auto **DisplayBalance** () const noexcept -> UnallocatedCString=0

  Returns the balance of the account as a formatted string for display in the UI.
- virtual auto **DisplayUnit** () const noexcept -> UnallocatedCString=0

  Returns the unit type of the account ("bitcoin" etc) as a formatted string for display in the UI.
- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns display name for the account.
- virtual auto **NotaryID** () const noexcept -> UnallocatedCString=0

  Returns the NotaryID for the account when relevant. (For off-chain accounts).
- virtual auto **NotaryName** () const noexcept -> UnallocatedCString=0

  Returns the display name for the account's notary, when relevant. (For off-chain accounts).
- virtual auto **Type** () const noexcept -> AccountType=0

  Returns the account type. (Issuer account, off-chain account, or blockchain account).
- virtual auto **Unit** () const noexcept -> UnitType=0

  Returns the unit type of the account as an enum.

6.8.1 Detailed Description

AccountTreeItem is the tree item that represents a single account, as seen from the AccountTree model. Each row in the AccountTree is an AccountCurrency, which contains all of the accounts of a given unit type. Each row in the AccountCurrency model contains an AccountTreeItem representing each of the accounts in the wallet of that currency type. For example, AccountTree may contain an AccountCurrency row for Bitcoin and an AccountCurrency row for Ethereum. The row for Bitcoin contains a list of Bitcoin accounts, and the row for Ethereum contains a list of Ethereum accounts (each represented as an AccountTreeItem).

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/AccountTreeItem.hpp

6.9 opentxs::api::session::Activity Class Reference

Public Member Functions

- virtual auto **AddBlockchainTransaction** (const blockchain::block::bitcoin::Transaction &transaction) const noexcept -> bool=0
- virtual auto **AddPaymentEvent** (const identifier::Nym &nymID, const Identifier &threadID, const otx::client::↩ StorageBox type, const Identifier &itemID, const Identifier &workflowID, Time time) const noexcept -> bool=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Activity &=0
- virtual auto Mail (const identifier::Nym &nym, const otx::client::StorageBox box) const noexcept -> Object↩ List=0
- virtual auto MailRemove (const identifier::Nym &nym, const Identifier &id, const otx::client::StorageBox box) const noexcept -> bool=0

- virtual auto MailText (const identifier::Nym &nym, const Identifier &id, const otx::client::StorageBox &box, const PasswordPrompt &reason) const noexcept -> std::shared_future< UnallocatedCString >=0
- virtual auto MarkRead (const identifier::Nym &nymId, const Identifier &threadId, const Identifier &itemId) const noexcept -> bool=0
- virtual auto MarkUnread (const identifier::Nym &nymId, const Identifier &threadId, const Identifier &itemId) const noexcept -> bool=0
- virtual auto PaymentText (const identifier::Nym &nym, const UnallocatedCString &id, const Unallocated↩ CString &workflow) const noexcept -> std::shared_ptr< const UnallocatedCString >=0
- virtual auto PreloadActivity (const identifier::Nym &nymID, const std::size_t count, const PasswordPrompt &reason) const noexcept -> void=0
- virtual auto PreloadThread (const identifier::Nym &nymID, const Identifier &threadID, const std::size_t start, const std::size_t count, const PasswordPrompt &reason) const noexcept -> void=0
- virtual OPENTXS_NO_EXPORT auto **Thread** (const identifier::Nym &nymID, const Identifier &threadID, proto::StorageThread &serialized) const noexcept -> bool=0
- virtual auto **Thread** (const identifier::Nym &nymID, const Identifier &threadID, AllocateOutput output) const noexcept -> bool=0
- virtual auto Threads (const identifier::Nym &nym, const bool unreadOnly=false) const noexcept -> Object↩ List=0
- virtual auto UnreadCount (const identifier::Nym &nym) const noexcept -> std::size_t=0
- virtual auto ThreadPublisher (const identifier::Nym &nym) const noexcept -> UnallocatedCString=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Activity &=0

6.9.1 Member Function Documentation

6.9.1.1 Mail()

```
virtual auto opentxs::api::session::Activity::Mail (
            const identifier::Nym & nym,
            const otx::client::StorageBox box ) const -> ObjectList  [pure virtual], [noexcept]
```

Obtain a list of mail objects in a specified box

Parameters

in	*nym*	the identifier of the nym who owns the mail box
in	*box*	the box to be listed

6.9.1.2 MailRemove()

```
virtual auto opentxs::api::session::Activity::MailRemove (
            const identifier::Nym & nym,
            const Identifier & id,
            const otx::client::StorageBox box ) const -> bool  [pure virtual], [noexcept]
```

Delete a mail object

Parameters

in	*nym*	the identifier of the nym who owns the mail box
in	*mail*	the mail object to be stored
in	*box*	the box from which to retrieve the mail object

Returns

The id of the stored message. The string will be empty if the mail object can not be stored.

6.9.1.3 MailText()

```
virtual auto opentxs::api::session::Activity::MailText (
            const identifier::Nym & nym,
            const Identifier & id,
            const otx::client::StorageBox & box,
            const PasswordPrompt & reason ) const -> std::shared_future< UnallocatedCString
> [pure virtual], [noexcept]
```

Retrieve the text from a message

Parameters

in	*nym*	the identifier of the nym who owns the mail box
in	*id*	the identifier of the mail object
in	*box*	the box from which to retrieve the mail object

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.9.1.4 MarkRead()

```
virtual auto opentxs::api::session::Activity::MarkRead (
            const identifier::Nym & nymId,
            const Identifier & threadId,
            const Identifier & itemId ) const -> bool  [pure virtual], [noexcept]
```

Mark a thread item as read

Parameters

in	*nymId*	the identifier of the nym who owns the thread
in	*threadId*	the thread containing the item to be marked
in	*itemId*	the identifier of the item to be marked read

Returns

False if the nym, thread, or item does not exist

6.9.1.5 MarkUnread()

```
virtual auto opentxs::api::session::Activity::MarkUnread (
            const identifier::Nym & nymId,
            const Identifier & threadId,
            const Identifier & itemId ) const -> bool  [pure virtual], [noexcept]
```

Mark a thread item as unread

Parameters

in	*nymId*	the identifier of the nym who owns the thread
in	*threadId*	the thread containing the item to be marked
in	*itemId*	the identifier of the item to be marked unread

Returns

False if the nym, thread, or item does not exist

6.9.1.6 PaymentText()

```
virtual auto opentxs::api::session::Activity::PaymentText (
            const identifier::Nym & nym,
            const UnallocatedCString & id,
            const UnallocatedCString & workflow ) const -> std::shared_ptr< const Unallocated↩
CString >  [pure virtual], [noexcept]
```

Summarize a payment workflow event in human-friendly test form

Parameters

in	*nym*	the identifier of the nym who owns the thread
in	*id*	the identifier of the payment item
in	*workflow*	the identifier of the payment workflow

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.9.1.7 PreloadActivity()

```
virtual auto opentxs::api::session::Activity::PreloadActivity (
            const identifier::Nym & nymID,
            const std::size_t count,
            const PasswordPrompt & reason ) const -> void  [pure virtual], [noexcept]
```

Asynchronously cache the most recent items in each of a nym's threads

Parameters

in	*nymID*	the identifier of the nym who owns the thread
in	*count*	the number of items to preload in each thread

6.9.1.8 PreloadThread()

```
virtual auto opentxs::api::session::Activity::PreloadThread (
            const identifier::Nym & nymID,
            const Identifier & threadID,
            const std::size_t start,
            const std::size_t count,
            const PasswordPrompt & reason ) const -> void  [pure virtual], [noexcept]
```

Asynchronously cache the items in an activity thread

Parameters

in	*nymID*	the identifier of the nym who owns the thread
in	*threadID*	the thread containing the items to be cached
in	*start*	the first item to be cached
in	*count*	the number of items to cache

6.9.1.9 ThreadPublisher()

```
virtual auto opentxs::api::session::Activity::ThreadPublisher (
            const identifier::Nym & nym ) const -> UnallocatedCString  [pure virtual], [noexcept]
```

Activity thread update notification

A subscribe socket can connect to this endpoint to receive ActivityThreadUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

6.9.1.10 Threads()

```
virtual auto opentxs::api::session::Activity::Threads (
            const identifier::Nym & nym,
            const bool unreadOnly = false ) const -> ObjectList  [pure virtual], [noexcept]
```

Obtain a list of thread ids for the specified nym

Parameters

in	*nym*	the identifier of the nym
in	*unreadOnly*	if true, only return threads with unread items

6.9.1.11 UnreadCount()

```
virtual auto opentxs::api::session::Activity::UnreadCount (
            const identifier::Nym & nym ) const -> std::size_t  [pure virtual], [noexcept]
```

Return the total number of unread thread items for a nym

Parameters

in	*nym↩ Id*	

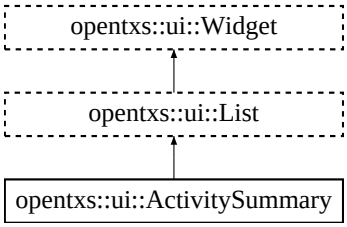
The documentation for this class was generated from the following file:

- include/opentxs/api/session/Activity.hpp

6.10 opentxs::ui::ActivitySummary Class Reference

```
#include <ActivitySummary.hpp>
```

Inheritance diagram for opentxs::ui::ActivitySummary:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ActivitySummaryItem >=0

  returns the first row, containing a valid ActivitySummaryItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ActivitySummaryItem >=0

  returns the next row, containing a valid ActivitySummaryItem or an empty smart pointer (if at end of list).

6.10.1 Detailed Description

This model manages the ActivitySummary. Each row is a different activity thread. For example, my chat session with Bob, and my chat session with Alice, constitute 2 rows in the ActivitySummary. Similar to a smart phone's list of past SMS conversations, each ActivitySummaryItem represents a different thread.

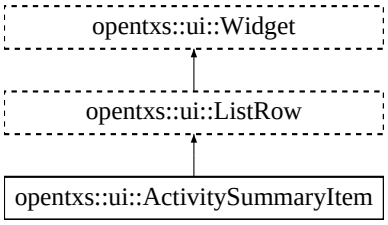
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ActivitySummary.hpp

6.11 opentxs::ui::ActivitySummaryItem Class Reference

`#include <ActivitySummaryItem.hpp>`

Inheritance diagram for opentxs::ui::ActivitySummaryItem:



Public Member Functions

- virtual auto **DisplayName** () const noexcept -> UnallocatedCString=0

  Returns the contact's display name for this thread.
- virtual auto **ImageURI** () const noexcept -> UnallocatedCString=0

  Returns the contact's image as a URI string.
- virtual auto **Text** () const noexcept -> UnallocatedCString=0

  Returns the display text (such as a message preview) for this thread.
- virtual auto **ThreadID** () const noexcept -> UnallocatedCString=0

  Returns the thread ID.
- virtual auto **Timestamp** () const noexcept -> Time=0

  Returns the timestamp of the most recent update to this thread.
- virtual auto **Type** () const noexcept -> otx::client::StorageBox=0

  Returns the thread type as an enum.

6.11.1 Detailed Description

ActivitySummaryItem is a high-level summary of a single conversational thread, meant to appear in a list of other recently active threads. Similar to the chat history on a smart phone, ActivitySummaryItem represents a single one of the conversational threads in that history.

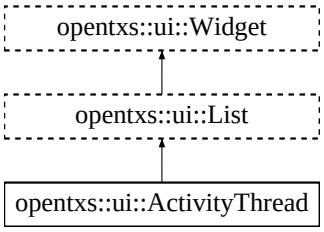
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ActivitySummaryItem.hpp

6.12 opentxs::ui::ActivityThread Class Reference

`#include <ActivityThread.hpp>`

Inheritance diagram for opentxs::ui::ActivityThread:



Public Member Functions

- virtual auto **CanMessage** () const noexcept -> bool=0

  Boolean value showing whether or not this contact can be messaged.
- virtual auto **DisplayName** () const noexcept -> UnallocatedCString=0

  Returns the display name for this thread (usually the name of the Contact).
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ActivityThreadItem >=0

  returns the first row, containing a valid ActivityThreadItem or an empty smart pointer (if list is empty).
- virtual auto **GetDraft** () const noexcept -> UnallocatedCString=0

  Returns the current draft (contains the draft of the newest outgoing message, not yet sent).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ActivityThreadItem >=0

  returns the next row, containing a valid ActivityThreadItem or an empty smart pointer (if at end of list).
- virtual auto **Participants** () const noexcept -> UnallocatedCString=0

  Returns a string containing the participants in this thread.
- virtual auto PaymentCode (const UnitType currency) const noexcept -> UnallocatedCString=0
- virtual auto **SendDraft** () const noexcept -> bool=0

  Sends the current draft.
- virtual auto **SetDraft** (const UnallocatedCString &draft) const noexcept -> bool=0

  Whenever the user makes edits to his newest unsent message, save the latest version here.
- virtual auto **ThreadID** () const noexcept -> UnallocatedCString=0

  Returns the ID for this thread.

Pay

Parameters

amount	*The amount being sent.*
sourceAccount	*The account where the funds are drawn from.*
memo	*Optional memo for the outgoing payment.*
type	*Type of payment being sent.*

Returns

Bool indicating whether payment was successfully sent.

- virtual auto **Pay** (const UnallocatedCString &amount, const Identifier &sourceAccount, const UnallocatedCString &memo="", const otx::client::PaymentType type=otx::client::PaymentType::Cheque) const noexcept -> bool=0
- virtual auto **Pay** (const Amount amount, const Identifier &sourceAccount, const UnallocatedCString &memo="", const otx::client::PaymentType type=otx::client::PaymentType::Cheque) const noexcept -> bool=0

6.12.1 Detailed Description

This model manages the ActivityThread between the user and one of his contacts. Each row in ActivityThread is a different ActivityThreadItem (chat message, incoming transaction, etc). This class is also a convenient way to grab relevant information about the current thread, such as the participants or the draft. Includes functionality for directly sending a payment inside the current thread.

6.12.2 Member Function Documentation

6.12.2.1 PaymentCode()

```
virtual auto opentxs::ui::ActivityThread::PaymentCode (
            const UnitType currency ) const -> UnallocatedCString  [pure virtual], [noexcept]
```

Returns the payment code for the contact relevant to this thread.

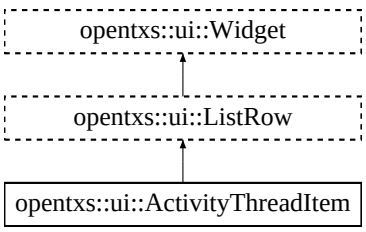
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ActivityThread.hpp

6.13 opentxs::ui::ActivityThreadItem Class Reference

```
#include <ActivityThreadItem.hpp>
```

Inheritance diagram for opentxs::ui::ActivityThreadItem:



Public Member Functions

- virtual auto **Amount** () const noexcept -> opentxs::Amount=0

  Returns the amount of the thread item, if relevant, as an Amount object.
- virtual auto **Deposit** () const noexcept -> bool=0

  Boolean value showing whether or not this item is a deposit to a server.
- virtual auto **DisplayAmount** () const noexcept -> UnallocatedCString=0

  Returns the amount of this thread item, if relevant, as a formatted string for display in the UI.
- virtual auto **From** () const noexcept -> UnallocatedCString=0

  Returns a string containing the ID of the sender.
- virtual auto **Loading** () const noexcept -> bool=0

  Boolean value showing whether or not this item is still in the process of loading.
- virtual auto **MarkRead** () const noexcept -> bool=0

  Boolean value showing whether or not this item is marked as read.
- virtual auto **Memo** () const noexcept -> UnallocatedCString=0

  Returns the memo, if relevant (for transactions).
- virtual auto **Outgoing** () const noexcept -> bool=0

  Boolean value showing whether or not this item is outgoing.
- virtual auto **Pending** () const noexcept -> bool=0

  Boolean value showing whether or not this item is pending.
- virtual auto **Text** () const noexcept -> UnallocatedCString=0

  Returns the main display text for this thread item.
- virtual auto **Timestamp** () const noexcept -> Time=0

  Returns the timestamp for this thread item.
- virtual auto **Type** () const noexcept -> otx::client::StorageBox=0

  Returns the type for this item. (Message, incoming cheque, outgoing blockchain transfer, etc).

6.13.1 Detailed Description

ActivityThreadItem is an individual chat message (or payment notice) as it appears inside the ActivityThread. For example, if Alice has sent 2 chat messages to Bob, then both users will see those messages as 2 individual ActivityThreadItems that appear as 2 rows inside an ActivityThread.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ActivityThreadItem.hpp

6.14 opentxs::api::network::Asio Class Reference

`#include <Asio.hpp>`

Public Types

- using **Endpoint** = opentxs::network::asio::Endpoint
- using **Socket** = opentxs::network::asio::Socket
- using **Resolved** = UnallocatedVector< Endpoint >
- using **AcceptCallback** = std::function< void(Socket &&)>

Public Member Functions

- auto Accept (const Endpoint &endpoint, AcceptCallback cb) const noexcept -> bool
- auto **Close** (const Endpoint &endpoint) const noexcept -> bool
- auto **GetPublicAddress4** () const noexcept -> std::shared_future< OTData >
- auto **GetPublicAddress6** () const noexcept -> std::shared_future< OTData >
- OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> internal::Asio &
- auto MakeSocket (const Endpoint &endpoint) const noexcept -> Socket
- auto NotificationEndpoint () const noexcept -> const char *
- auto **Resolve** (std::string_view server, std::uint16_t port) const noexcept -> Resolved
- OPENTXS_NO_EXPORT auto **Init** () noexcept -> void
- OPENTXS_NO_EXPORT auto **Shutdown** () noexcept -> void
- OPENTXS_NO_EXPORT **Asio** (const opentxs::network::zeromq::Context &zmq) noexcept

6.14.1 Detailed Description

The api::network::Asio API contains functions used for networking via Boost::ASIO.

6.14.2 Member Function Documentation

6.14.2.1 Accept()

```
auto opentxs::api::network::Asio::Accept (
            const Endpoint & endpoint,
            AcceptCallback cb ) const -> bool  [noexcept]
```

Receive incoming tcp and udp connections

Calling this function will instruct the operating system to monitor a specified endpoint for incoming connection requests.

Once a connection request has been processed Asio will generate a socket and deliver it to the caller via the provided callback

Parameters

endpoint	the address / port which will be monitored for incoming connection requests.

Warning

The caller must ensure the lifetime of the endpoint lasts until Close() has been called

Parameters

cb	the callback function which will be executed to deliver a newly created socket once an incoming connection request has been received

Returns

true if the operating system accepts the request to set up incoming connection handling on the specified socket

6.14.2.2 MakeSocket()

```
auto opentxs::api::network::Asio::MakeSocket (
            const Endpoint & endpoint ) const -> Socket  [noexcept]
```

Construct a socket for outgoing tcp and udp connections

Parameters

endpoint	the address / port to which an outgoing connection will be created

Warning

The caller must ensure the lifetime of the endpoint exceeds the lifetime of the socket

6.14.2.3 NotificationEndpoint()

```
auto opentxs::api::network::Asio::NotificationEndpoint ( ) const -> const char *  [noexcept]
```

Endpoint for asio to zeromq message routing

This class maintained a zeromq router socket which is bound to the endpoint specified by this function.

After connecting to this endpoint with a zeromq dealer socket, callers should send an AsioRegister message as described in util/WorkType.hpp

The sequence of bytes received as the payload of the AsioRegister response is the value that must be provided to the asio::Socket::Connect and asio::Socket::Receive functions

The documentation for this class was generated from the following file:

- include/opentxs/api/network/Asio.hpp

6.15 opentxs::api::crypto::Asymmetric Class Reference

```
#include <Asymmetric.hpp>
```

Public Member Functions

- virtual auto **InstantiateKey** (const opentxs::crypto::key::asymmetric::Algorithm type, const Unallocated↩ CString &seedID, const opentxs::crypto::Bip32::Key &serialized, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **InstantiateKey** (const opentxs::crypto::key::asymmetric::Algorithm type, const Unallocated↩ CString &seedID, const opentxs::crypto::Bip32::Key &serialized, const opentxs::crypto::key::asymmetric::↩ Role role, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **InstantiateKey** (const opentxs::crypto::key::asymmetric::Algorithm type, const Unallocated↩ CString &seedID, const opentxs::crypto::Bip32::Key &serialized, const VersionNumber version, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **InstantiateKey** (const opentxs::crypto::key::asymmetric::Algorithm type, const Unallocated↩ CString &seedID, const opentxs::crypto::Bip32::Key &serialized, const opentxs::crypto::key::asymmetric::↩ Role role, const VersionNumber version, const PasswordPrompt &reason) const noexcept -> std::unique_↩ ptr< opentxs::crypto::key::HD >=0
- virtual auto **InstantiateSecp256k1Key** (const ReadView publicKey, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const ReadView publicKey, const opentxs::crypto::key↩ ::asymmetric::Role role, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs↩ ::crypto::key::Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const ReadView publicKey, const VersionNumber version, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const ReadView publicKey, const opentxs::crypto::key↩ ::asymmetric::Role role, const VersionNumber version, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const Secret &privateKey, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const Secret &privateKey, const opentxs::crypto::key::asymmetric↩ ::Role role, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::↩ Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const Secret &privateKey, const VersionNumber version, const PasswordPrompt &reason) const noexcept -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **InstantiateSecp256k1Key** (const Secret &privateKey, const opentxs::crypto::key::asymmetric↩ ::Role role, const VersionNumber version, const PasswordPrompt &reason) const noexcept -> std::unique↩ _ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Asymmetric &=0
- virtual auto **NewHDKey** (const UnallocatedCString &seedID, const Secret &seed, const opentxs::crypto↩ ::EcdsaCurve &curve, const opentxs::crypto::Bip32::Path &path, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **NewHDKey** (const UnallocatedCString &seedID, const Secret &seed, const opentxs::crypto::↩ EcdsaCurve &curve, const opentxs::crypto::Bip32::Path &path, const opentxs::crypto::key::asymmetric::Role role, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **NewHDKey** (const UnallocatedCString &seedID, const Secret &seed, const opentxs::crypto↩ ::EcdsaCurve &curve, const opentxs::crypto::Bip32::Path &path, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **NewHDKey** (const UnallocatedCString &seedID, const Secret &seed, const opentxs::crypto::↩ EcdsaCurve &curve, const opentxs::crypto::Bip32::Path &path, const opentxs::crypto::key::asymmetric::Role role, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs↩ ::crypto::key::HD >=0
- virtual auto **NewKey** (const opentxs::crypto::Parameters &params, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Asymmetric >=0
- virtual auto **NewKey** (const opentxs::crypto::Parameters &params, const opentxs::crypto::key::asymmetric↩ ::Role role, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Asymmetric >=0
- virtual auto **NewKey** (const opentxs::crypto::Parameters &params, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Asymmetric >=0

- virtual auto **NewKey** (const opentxs::crypto::Parameters &params, const opentxs::crypto::key::asymmetric↩ ::Role role, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Asymmetric >=0
- virtual auto **NewSecp256k1Key** (const UnallocatedCString &seedID, const Secret &seed, const opentxs↩ ::crypto::Bip32::Path &path, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto↩ ::key::Secp256k1 >=0
- virtual auto **NewSecp256k1Key** (const UnallocatedCString &seedID, const Secret &seed, const opentxs↩ ::crypto::Bip32::Path &path, const opentxs::crypto::key::asymmetric::Role role, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **NewSecp256k1Key** (const UnallocatedCString &seedID, const Secret &seed, const opentxs↩ ::crypto::Bip32::Path &path, const VersionNumber version, const PasswordPrompt &reason) const -> std↩ ::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **NewSecp256k1Key** (const UnallocatedCString &seedID, const Secret &seed, const opentxs↩ ::crypto::Bip32::Path &path, const opentxs::crypto::key::asymmetric::Role role, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Asymmetric &=0

6.15.1 Detailed Description

The api::crypto::Asymmetric API is used for instantiating asymmetric keys and related crypto objects.

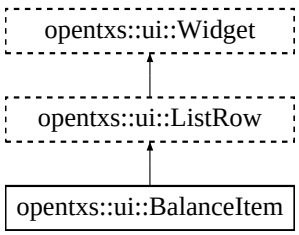
The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Asymmetric.hpp

6.16 opentxs::ui::BalanceItem Class Reference

`#include <BalanceItem.hpp>`

Inheritance diagram for opentxs::ui::BalanceItem:



Public Member Functions

- virtual auto **Amount** () const noexcept -> opentxs::Amount=0

  Returns the balance item's amount, as an Amount object.
- virtual auto **Confirmations** () const noexcept -> int=0

  Returns the number of confirmations for the associated blockchain transaction, if relevant.
- virtual auto **Contacts** () const noexcept -> UnallocatedVector< UnallocatedCString >=0

  Returns a vector of strings containing Contacts relevant to this balance item.
- virtual auto **DisplayAmount** () const noexcept -> UnallocatedCString=0

  Returns the amount of this balance item as a string formatted for display in the UI.

- virtual auto **Memo** () const noexcept -> UnallocatedCString=0

  Returns the memo for this balance item.
- virtual auto **Text** () const noexcept -> UnallocatedCString=0

  Returns the main display text for this balance item.
- virtual auto **Timestamp** () const noexcept -> Time=0

  Returns the timestamp of this balance item, as a Time object.
- virtual auto **Type** () const noexcept -> otx::client::StorageBox=0

  Returns the type of this balance item as an enum.
- virtual auto **UUID** () const noexcept -> UnallocatedCString=0

  Returns the UUID of this balance item.
- virtual auto **Workflow** () const noexcept -> UnallocatedCString=0

  Returns the workflow of this balance item.

6.16.1 Detailed Description

This model contains a Balance Item, representing a single ledger entry for a specific account. The AccountActivity class contains a list of Balance Items.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BalanceItem.hpp

6.17 opentxs::api::crypto::Blockchain Class Reference

`#include <Blockchain.hpp>`

Public Types

- using **Chain** = opentxs::blockchain::Type
- using **Key** = opentxs::blockchain::crypto::Key
- using **Style** = opentxs::blockchain::crypto::AddressStyle
- using **Subchain** = opentxs::blockchain::crypto::Subchain
- using **DecodedAddress** = std::tuple< OTData, Style, UnallocatedSet< Chain >, bool >
- using **ContactList** = UnallocatedSet< OTIdentifier >
- using **Txid** = opentxs::blockchain::block::Txid
- using **TxidHex** = UnallocatedCString
- using **PatternID** = opentxs::blockchain::PatternID
- using **AccountData** = std::pair< Chain, OTNymID >

Public Member Functions

- virtual auto **Account** (const identifier::Nym &nymID, const Chain chain) const noexcept(false) -> const opentxs::blockchain::crypto::Account &=0

  Throws std::runtime_error if chain is invalid.
- virtual auto **AccountList** (const identifier::Nym &nymID) const noexcept -> UnallocatedSet< OTIdentifier >=0
- virtual auto **AccountList** (const Chain chain) const noexcept -> UnallocatedSet< OTIdentifier >=0
- virtual auto **AccountList** () const noexcept -> UnallocatedSet< OTIdentifier >=0
- virtual auto **ActivityDescription** (const identifier::Nym &nym, const Identifier &thread, const Unallocated↩ CString &threadItemID) const noexcept -> UnallocatedCString=0
- virtual auto **ActivityDescription** (const identifier::Nym &nym, const Chain chain, const opentxs↩ ::blockchain::block::bitcoin::Transaction &transaction) const noexcept -> UnallocatedCString=0
- virtual auto **AssignContact** (const identifier::Nym &nymID, const Identifier &accountID, const Subchain subchain, const Bip32Index index, const Identifier &label) const noexcept -> bool=0
- virtual auto **AssignLabel** (const identifier::Nym &nymID, const Identifier &accountID, const Subchain subchain, const Bip32Index index, const UnallocatedCString &label) const noexcept -> bool=0
- virtual auto **AssignTransactionMemo** (const TxidHex &id, const UnallocatedCString &label) const noexcept -> bool=0
- virtual auto **CalculateAddress** (const opentxs::blockchain::Type chain, const opentxs::blockchain::crypto↩ ::AddressStyle format, const Data &pubkey) const noexcept -> UnallocatedCString=0
- virtual auto **Confirm** (const Key key, const opentxs::blockchain::block::Txid &tx) const noexcept -> bool=0
- virtual auto **DecodeAddress** (const UnallocatedCString &encoded) const noexcept -> DecodedAddress=0
- virtual auto **EncodeAddress** (const Style style, const Chain chain, const Data &data) const noexcept -> UnallocatedCString=0
- virtual auto **GetKey** (const Key &id) const noexcept(false) -> const opentxs::blockchain::crypto::Element &=0

  Throws std::out_of_range if the specified key does not exist.
- virtual auto **HDSubaccount** (const identifier::Nym &nymID, const Identifier &accountID) const noexcept(false) -> const opentxs::blockchain::crypto::HD &=0

  Throws std::out_of_range if the specified account does not exist.
- virtual auto **IndexItem** (const ReadView bytes) const noexcept -> PatternID=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const crypto::internal::Blockchain &=0
- virtual auto **LoadTransactionBitcoin** (const Txid &id) const noexcept -> std::unique_ptr< const opentxs↩ ::blockchain::block::bitcoin::Transaction >=0
- virtual auto **LoadTransactionBitcoin** (const TxidHex &id) const noexcept -> std::unique_ptr< const opentxs::blockchain::block::bitcoin::Transaction >=0
- virtual auto **LookupAccount** (const Identifier &id) const noexcept -> AccountData=0
- virtual auto **LookupContacts** (const UnallocatedCString &address) const noexcept -> ContactList=0
- virtual auto **LookupContacts** (const Data &pubkeyHash) const noexcept -> ContactList=0
- virtual auto **NewHDSubaccount** (const identifier::Nym &nymID, const opentxs::blockchain::crypto::↩ HDProtocol standard, const Chain chain, const PasswordPrompt &reason) const noexcept -> OTIdentifier=0
- virtual auto **NewHDSubaccount** (const identifier::Nym &nymID, const opentxs::blockchain::crypto::↩ HDProtocol standard, const Chain derivationChain, const Chain targetChain, const PasswordPrompt &reason) const noexcept -> OTIdentifier=0
- virtual auto **NewPaymentCodeSubaccount** (const identifier::Nym &nymID, const opentxs::PaymentCode &local, const opentxs::PaymentCode &remote, const ReadView &view, const Chain chain, const Password↩ Prompt &reason) const noexcept -> OTIdentifier=0
- virtual auto **Owner** (const Identifier &accountID) const noexcept -> const identifier::Nym &=0
- virtual auto **Owner** (const Key &key) const noexcept -> const identifier::Nym &=0
- virtual auto **PaymentCodeSubaccount** (const identifier::Nym &nymID, const Identifier &accountID) const noexcept(false) -> const opentxs::blockchain::crypto::PaymentCode &=0

  Throws std::out_of_range if the specified account does not exist.
- virtual auto **RecipientContact** (const Key &key) const noexcept -> OTIdentifier=0
- virtual auto **Release** (const Key key) const noexcept -> bool=0

- virtual auto **SenderContact** (const Key &key) const noexcept -> OTIdentifier=0
- virtual auto **SubaccountList** (const identifier::Nym &nymID, const Chain chain) const noexcept -> UnallocatedSet< OTIdentifier >=0
- virtual auto **Unconfirm** (const Key key, const opentxs::blockchain::block::Txid &tx, const Time time=Clock↩ ::now()) const noexcept -> bool=0
- virtual auto **Wallet** (const Chain chain) const noexcept(false) -> const opentxs::blockchain::crypto::Wallet &=0

  Throws std::runtime_error if chain is invalid.
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> crypto::internal::Blockchain &=0

Static Public Member Functions

- static auto **Bip44** (Chain chain) noexcept(false) -> Bip44Type
- static auto **Bip44Path** (Chain chain, const UnallocatedCString &seed, AllocateOutput destination) noexcept(false) -> bool

6.17.1 Detailed Description

The api::crypto::Blockchain API is used for accessing blockchain-related crypto functionality.

The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Blockchain.hpp

6.18 opentxs::api::network::Blockchain Class Reference

`#include <Blockchain.hpp>`

Public Types

- using **Chain** = opentxs::blockchain::Type
- using **Endpoints** = UnallocatedVector< UnallocatedCString >

Public Member Functions

- auto **AddSyncServer** (const UnallocatedCString &endpoint) const noexcept -> bool
- auto **ConnectedSyncServers** () const noexcept -> Endpoints
- auto **DeleteSyncServer** (const UnallocatedCString &endpoint) const noexcept -> bool
- auto **Disable** (const Chain type) const noexcept -> bool
- auto **Enable** (const Chain type, const UnallocatedCString &seednode="") const noexcept -> bool
- auto **EnabledChains** () const noexcept -> UnallocatedSet< Chain >
- auto **GetChain** (const Chain type) const noexcept(false) -> const opentxs::blockchain::node::Manager &

  throws std::out_of_range if chain has not been started
- auto **GetSyncServers** () const noexcept -> Endpoints
- OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> internal::Blockchain &
- auto **Start** (const Chain type, const UnallocatedCString &seednode="") const noexcept -> bool
- auto **StartSyncServer** (const UnallocatedCString &syncEndpoint, const UnallocatedCString &publicSync↩ Endpoint, const UnallocatedCString &updateEndpoint, const UnallocatedCString &publicUpdateEndpoint) const noexcept -> bool
- auto **Stop** (const Chain type) const noexcept -> bool
- OPENTXS_NO_EXPORT **Blockchain** (Imp *imp) noexcept

6.18.1 Detailed Description

The api::network::Blockchain API is used for accessing blockchain-related network functionality.

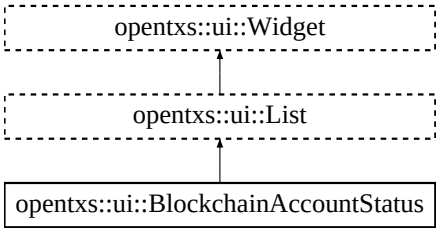
The documentation for this class was generated from the following file:

- include/opentxs/api/network/Blockchain.hpp

6.19 opentxs::ui::BlockchainAccountStatus Class Reference

`#include <BlockchainAccountStatus.hpp>`

Inheritance diagram for opentxs::ui::BlockchainAccountStatus:



Public Member Functions

- virtual auto **Chain** () const noexcept -> blockchain::Type=0

  Returns the chain type for this blockchain account.
- virtual auto **First** () const noexcept -> SharedPimpl< BlockchainSubaccountSource >=0

  returns the first row, containing a valid BlockchainSubaccountSource or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> SharedPimpl< BlockchainSubaccountSource >=0

  returns the next row, containing a valid BlockchainSubaccountSource or an empty smart pointer (if at end of list).
- virtual auto **Owner** () const noexcept -> const identifier::Nym &=0

  Returns the NymID of the owner of this account.

6.19.1 Detailed Description

This model contains the status for a single blockchain account. Each subaccount is a row in this model.

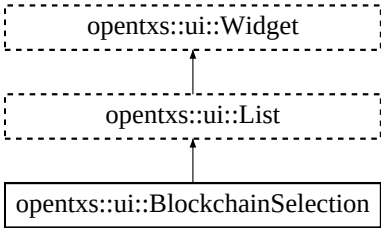
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainAccountStatus.hpp

6.20 opentxs::ui::BlockchainSelection Class Reference

`#include <BlockchainSelection.hpp>`

Inheritance diagram for opentxs::ui::BlockchainSelection:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::BlockchainSelectionItem >=0
  returns the first row, containing a valid BlockchainSelectionItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::BlockchainSelectionItem >=0
  returns the next row, containing a valid BlockchainSelectionItem or an empty smart pointer (if at end of list).
- virtual auto **Disable** (const blockchain::Type type) const noexcept -> bool=0
  This function can be used to disable a blockchain in the wallet. Returns success or failure.
- virtual auto **Enable** (const blockchain::Type type) const noexcept -> bool=0
  This function can be used to enable a blockchain in the wallet. Returns success or failure.

6.20.1 Detailed Description

The rows in this model each represent a blockchain supported by the wallet. Each chain can be individually enabled or disabled.

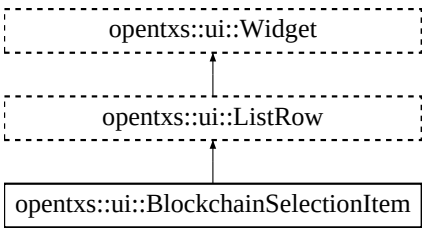
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainSelection.hpp

6.21 opentxs::ui::BlockchainSelectionItem Class Reference

`#include <BlockchainSelectionItem.hpp>`

Inheritance diagram for opentxs::ui::BlockchainSelectionItem:



Public Member Functions

- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns the display name of this blockchain.
- virtual auto **IsEnabled** () const noexcept -> bool=0

  Returns boolean indicating whether or not this blockchain is enabled.
- virtual auto **IsTestnet** () const noexcept -> bool=0

  Returns boolean indicating whether or not this blockchain is a testnet.
- virtual auto **Type** () const noexcept -> blockchain::Type=0

  Returns enum containing the blockchain type.

6.21.1 Detailed Description

This model represents a single blockchain. It is a single row from the BlockchainSelection model.

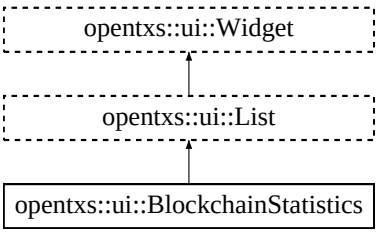
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainSelectionItem.hpp

6.22 opentxs::ui::BlockchainStatistics Class Reference

`#include <BlockchainStatistics.hpp>`

Inheritance diagram for opentxs::ui::BlockchainStatistics:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::BlockchainStatisticsItem >=0

  returns the first row, containing a valid BlockchainStatisticsItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::BlockchainStatisticsItem >=0

  returns the next row, containing a valid BlockchainStatisticsItem or an empty smart pointer (if at end of list).

6.22.1 Detailed Description

The rows in this model each represent a single blockchain and its related wallet statistics.

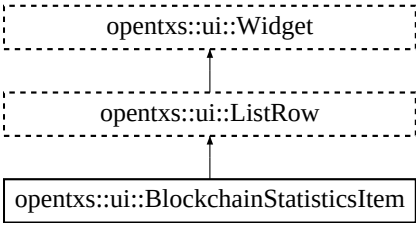
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainStatistics.hpp

6.23 opentxs::ui::BlockchainStatisticsItem Class Reference

`#include <BlockchainStatisticsItem.hpp>`

Inheritance diagram for opentxs::ui::BlockchainStatisticsItem:



Public Types

- using **Position** = blockchain::block::Height

Public Member Functions

- virtual auto **ActivePeers** () const noexcept -> std::size_t=0

  Returns the number of active peers for this blockchain.
- virtual auto **Balance** () const noexcept -> UnallocatedCString=0

  Returns the current balance for this blockchain.
- virtual auto **BlockDownloadQueue** () const noexcept -> std::size_t=0

  Returns the number of blocks currently in the block download queue.
- virtual auto **Chain** () const noexcept -> blockchain::Type=0

  Returns the blockchain type (Bitcoin, Ethereum, etc).
- virtual auto **ConnectedPeers** () const noexcept -> std::size_t=0

  Returns the number of connected peers for this blockchain.
- virtual auto **Filters** () const noexcept -> Position=0

  Returns the number of filters downloaded so far for this blockchain.
- virtual auto **Headers** () const noexcept -> Position=0

  Returns the number of block headers downloaded so far for this blockchain.
- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns the display name for this blockchain.

6.23.1 Detailed Description

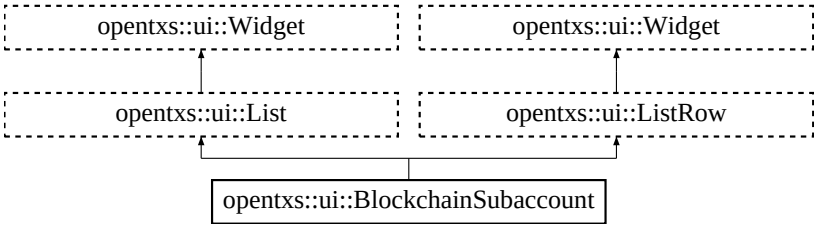
This model represents the statistics for a single blockchain in the wallet. There is a model like this for each row in the BlockchainStatistics model.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainStatisticsItem.hpp

6.24 opentxs::ui::BlockchainSubaccount Class Reference

Inheritance diagram for opentxs::ui::BlockchainSubaccount:



Public Member Functions

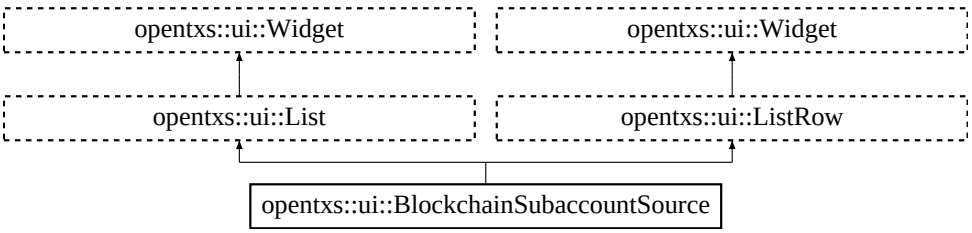
- virtual auto **First** () const noexcept -> SharedPimpl< BlockchainSubchain >=0
- virtual auto **Name** () const noexcept -> UnallocatedCString=0
- virtual auto **Next** () const noexcept -> SharedPimpl< BlockchainSubchain >=0
- virtual auto **SubaccountID** () const noexcept -> const Identifier &=0

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainSubaccount.hpp

6.25 opentxs::ui::BlockchainSubaccountSource Class Reference

Inheritance diagram for opentxs::ui::BlockchainSubaccountSource:



Public Member Functions

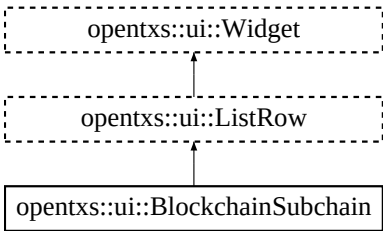
- virtual auto **First** () const noexcept -> SharedPimpl< BlockchainSubaccount >=0
- virtual auto **Name** () const noexcept -> UnallocatedCString=0
- virtual auto **Next** () const noexcept -> SharedPimpl< BlockchainSubaccount >=0
- virtual auto **SourceID** () const noexcept -> const Identifier &=0
- virtual auto **Type** () const noexcept -> blockchain::crypto::SubaccountType=0

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainSubaccountSource.hpp

6.26 opentxs::ui::BlockchainSubchain Class Reference

Inheritance diagram for opentxs::ui::BlockchainSubchain:



Public Member Functions

- virtual auto **Name** () const noexcept -> UnallocatedCString=0
- virtual auto **Progress** () const noexcept -> UnallocatedCString=0
- virtual auto **Type** () const noexcept -> blockchain::crypto::Subchain=0

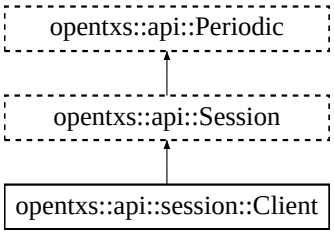
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/BlockchainSubchain.hpp

6.27 opentxs::api::session::Client Class Reference

`#include <Client.hpp>`

Inheritance diagram for opentxs::api::session::Client:



Public Member Functions

- virtual auto **Activity** () const -> const session::Activity &=0

  Returns the session's Activities.
- virtual auto **Contacts** () const -> const api::session::Contacts &=0

  Returns the session's Contacts.
- virtual OPENTXS_NO_EXPORT auto **InternalClient** () const noexcept -> const internal::Client &=0
- virtual auto **OTX** () const -> const session::OTX &=0

  Returns the OTX API for this session.
- virtual auto **UI** () const -> const session::UI &=0

  Returns the UI API for this session.
- virtual auto **Workflow** () const -> const session::Workflow &=0

  Returns the Workflow API for this session.
- virtual auto **ZMQ** () const -> const network::ZMQ &=0

  Returns the ZMQ API for this session. For message passing.
- virtual OPENTXS_NO_EXPORT auto **InternalClient** () noexcept -> internal::Client &=0

6.27.1 Detailed Description

Returns various API handles related to a client session.

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Client.hpp

6.28 opentxs::api::crypto::Config Class Reference

`#include <Config.hpp>`

Public Member Functions

- virtual OPENTXS_NO_EXPORT auto **InternalConfig** () const noexcept -> const internal::Config &=0
- virtual auto **IterationCount** () const -> std::uint32_t=0
- virtual auto **SymmetricSaltSize** () const -> std::uint32_t=0
- virtual auto **SymmetricKeySize** () const -> std::uint32_t=0
- virtual auto **SymmetricKeySizeMax** () const -> std::uint32_t=0
- virtual auto **SymmetricIvSize** () const -> std::uint32_t=0
- virtual auto **SymmetricBufferSize** () const -> std::uint32_t=0
- virtual auto **PublicKeysize** () const -> std::uint32_t=0
- virtual auto **PublicKeysizeMax** () const -> std::uint32_t=0
- virtual OPENTXS_NO_EXPORT auto **InternalConfig** () noexcept -> internal::Config &=0

6.28.1 Detailed Description

The api::crypto::Config API is used for accessing crypto-specific configuration information.

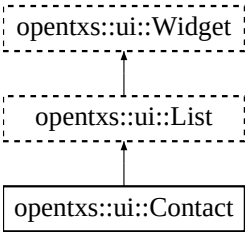
The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Config.hpp

6.29 opentxs::ui::Contact Class Reference

`#include <Contact.hpp>`

Inheritance diagram for opentxs::ui::Contact:



Public Member Functions

- virtual auto **ContactID** () const noexcept -> UnallocatedCString=0

  Returns the ContactID for this contact.
- virtual auto **DisplayName** () const noexcept -> UnallocatedCString=0

  Returns the display label for this contact.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactSection >=0

  Returns the first contact section.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactSection >=0

  Returns the next contact section.
- virtual auto **PaymentCode** () const noexcept -> UnallocatedCString=0

  Returns the payment code for this contact.

6.29.1 Detailed Description

This model represents a single contact. Each row is a ContactSection containing metadata about this contact.

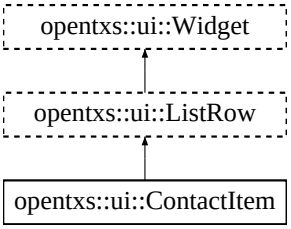
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/Contact.hpp

6.30 opentxs::ui::ContactItem Class Reference

`#include <ContactItem.hpp>`

Inheritance diagram for opentxs::ui::ContactItem:



Public Member Functions

- virtual auto **ClaimID** () const noexcept -> UnallocatedCString=0

  Returns the claim ID for this claim.
- virtual auto **IsActive** () const noexcept -> bool=0

  Returns a boolean indicating whether or not this claim is active.
- virtual auto **IsPrimary** () const noexcept -> bool=0

  Returns a boolean indicating whether or not this is the primary claim.
- virtual auto **Value** () const noexcept -> UnallocatedCString=0

  Returns the actual contains of the claim as a string.

6.30.1 Detailed Description

Each row in the ContactSubsection model is a ContactItem. This ContactItem model represents a specific claim about a Contact.

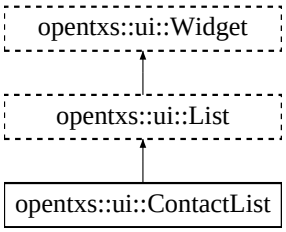
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ContactItem.hpp

6.31 opentxs::ui::ContactList Class Reference

`#include <ContactList.hpp>`

Inheritance diagram for opentxs::ui::ContactList:



Public Member Functions

- virtual auto **AddContact** (const UnallocatedCString &label, const UnallocatedCString &paymentCode="", const UnallocatedCString &nymID="") const noexcept -> UnallocatedCString=0

  Adds a Contact to the wallet.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactListItem >=0

  returns the first row, containing a valid ContactListItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactListItem >=0

  returns the next row, containing a valid ContactListItem or an empty smart pointer (if at end of list).

6.31.1 Detailed Description

This model manages a set of rows containing the Contacts in the wallet. It also contains a method for adding new contacts.

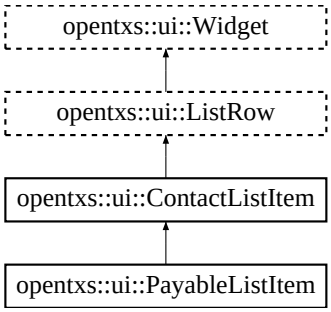
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ContactList.hpp

6.32 opentxs::ui::ContactListItem Class Reference

`#include <ContactListItem.hpp>`

Inheritance diagram for opentxs::ui::ContactListItem:



Public Member Functions

- virtual auto **ContactID** () const noexcept -> UnallocatedCString=0

  Returns this contact's Contact ID.
- virtual auto **DisplayName** () const noexcept -> UnallocatedCString=0

  Returns the display name for this Contact.
- virtual auto **ImageURI** () const noexcept -> UnallocatedCString=0

  Returns the image URI for this contact.
- virtual auto **Section** () const noexcept -> UnallocatedCString=0

  Returns the section for this contact.

6.32.1 Detailed Description

This model represents a single ContactListItem from a row in the ContactList model.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ContactListItem.hpp

6.33 opentxs::api::session::Contacts Class Reference

Public Member Functions

- virtual auto **Contact** (const Identifier &id) const -> std::shared_ptr< const opentxs::Contact >=0
- virtual auto ContactID (const identifier::Nym &nymID) const -> OTIdentifier=0
- virtual auto **ContactList** () const -> ObjectList=0
- virtual auto **ContactName** (const Identifier &contactID) const -> UnallocatedCString=0
- virtual auto **ContactName** (const Identifier &contactID, UnitType currencyHint) const -> Unallocated↩ CString=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Contacts &=0
- virtual auto **Merge** (const Identifier &parent, const Identifier &child) const -> std::shared_ptr< const opentxs::Contact >=0

- virtual auto **NewContact** (const UnallocatedCString &label) const -> std::shared_ptr< const opentxs::↩ Contact >=0
- virtual auto **NewContact** (const UnallocatedCString &label, const identifier::Nym &nymID, const Payment↩ Code &paymentCode) const -> std::shared_ptr< const opentxs::Contact >=0
- virtual auto **NewContactFromAddress** (const UnallocatedCString &address, const UnallocatedCString &label, const opentxs::blockchain::Type currency) const -> std::shared_ptr< const opentxs::Contact >=0
- virtual auto NymToContact (const identifier::Nym &nymID) const -> OTIdentifier=0
- virtual auto PaymentCodeToContact (const PaymentCode &code, const opentxs::blockchain::Type currency) const -> OTIdentifier=0
- virtual auto **PaymentCodeToContact** (const UnallocatedCString &code, const opentxs::blockchain::Type currency) const -> OTIdentifier=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Contacts &=0

6.33.1 Member Function Documentation

6.33.1.1 ContactID()

```
virtual auto opentxs::api::session::Contacts::ContactID (
            const identifier::Nym & nymID ) const -> OTIdentifier  [pure virtual]
```

Returns the contact ID for a nym, if it exists

6.33.1.2 NymToContact()

```
virtual auto opentxs::api::session::Contacts::NymToContact (
            const identifier::Nym & nymID ) const -> OTIdentifier  [pure virtual]
```

Returns an existing contact ID if it exists, or creates a new one

6.33.1.3 PaymentCodeToContact()

```
virtual auto opentxs::api::session::Contacts::PaymentCodeToContact (
            const PaymentCode & code,
            const opentxs::blockchain::Type currency ) const -> OTIdentifier  [pure virtual]
```

Returns an existing contact ID if it exists, or creates a new one

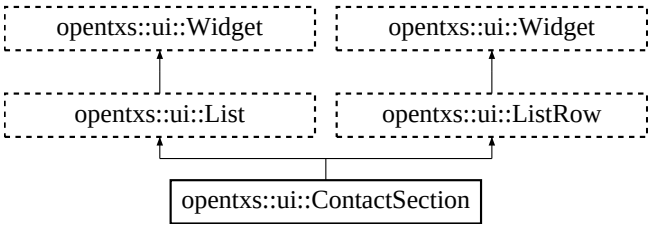
The documentation for this class was generated from the following file:

- include/opentxs/api/session/Contacts.hpp

6.34 opentxs::ui::ContactSection Class Reference

`#include <ContactSection.hpp>`

Inheritance diagram for opentxs::ui::ContactSection:



Public Member Functions

- virtual auto **Name** (const UnallocatedCString &lang) const noexcept -> UnallocatedCString=0

  Returns the section name.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactSubsection >=0

  Returns the first contact subsection.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactSubsection >=0

  Returns the next contact subsection.
- virtual auto **Type** () const noexcept -> identity::wot::claim::SectionType=0

  Returns the section type as an enum.

6.34.1 Detailed Description

This model represents a section of meta-data for a specific contact. Each row is a ContactSubsection containing metadata about this contact.

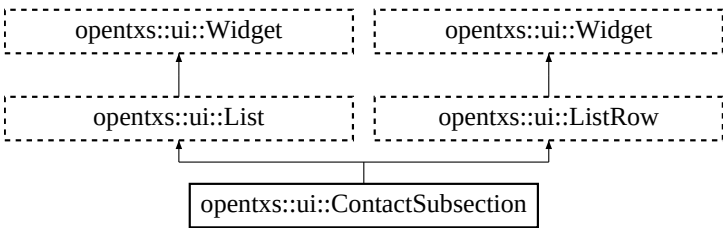
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ContactSection.hpp

6.35 opentxs::ui::ContactSubsection Class Reference

`#include <ContactSubsection.hpp>`

Inheritance diagram for opentxs::ui::ContactSubsection:



Public Member Functions

- virtual auto **Name** (const UnallocatedCString &lang) const noexcept -> UnallocatedCString=0

  Returns the name of the subsection.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactItem >=0

  Returns the first ContactItem row.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactItem >=0

  Returns the next ContactItem row.
- virtual auto **Type** () const noexcept -> identity::wot::claim::ClaimType=0

  Returns the claim type as a ClaimType enum.

6.35.1 Detailed Description

This model represents a subsection of meta-data for a ContactSection of a specific Contact. Each row is a ContactItem containing metadata about this contact.

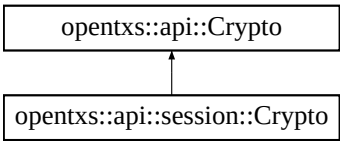
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ContactSubsection.hpp

6.36 opentxs::api::Crypto Class Reference

`#include <Crypto.hpp>`

Inheritance diagram for opentxs::api::Crypto:



Public Member Functions

- virtual auto **BIP32** () const noexcept -> const opentxs::crypto::Bip32 &=0
- virtual auto **BIP39** () const noexcept -> const opentxs::crypto::Bip39 &=0
- virtual auto **Config** () const noexcept -> const crypto::Config &=0

  Returns a handle to the Config API from api::Crypto.
- virtual auto **Encode** () const noexcept -> const crypto::Encode &=0
- virtual auto **Hash** () const noexcept -> const crypto::Hash &=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Crypto &=0
- virtual auto **Util** () const noexcept -> const crypto::Util &=0

  Returns a handle to the Util API from api::Crypto.
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Crypto &=0

6.36.1 Detailed Description

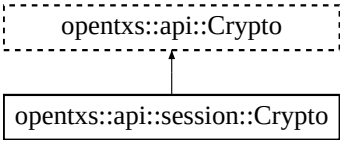
The api::crypto API is used for accessing high-level crypto-specific functionality like Encode or Hash. It's also used for accessing lower-level APIs such as Config or Util.

The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Crypto.hpp

6.37 opentxs::api::session::Crypto Class Reference

Inheritance diagram for opentxs::api::session::Crypto:



Public Member Functions

- virtual auto **Asymmetric** () const noexcept -> const crypto::Asymmetric &=0
- virtual auto **Blockchain** () const noexcept -> const crypto::Blockchain &=0
- virtual OPENTXS_NO_EXPORT auto **InternalSession** () const noexcept -> const internal::Crypto &=0
- virtual auto **Seed** () const noexcept -> const crypto::Seed &=0
- virtual auto **Symmetric** () const noexcept -> const crypto::Symmetric &=0
- virtual OPENTXS_NO_EXPORT auto **InternalSession** () noexcept -> internal::Crypto &=0

6.37.1 Constructor & Destructor Documentation

6.37.1.1 ~Crypto()

```
OPENTXS_NO_EXPORT opentxs::api::session::Crypto::~Crypto ( )  [override], [virtual], [default]
```

Reimplemented from opentxs::api::Crypto.

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Crypto.hpp

6.38 opentxs::api::network::Dht Class Reference

```
#include <Dht.hpp>
```

Public Member Functions

- virtual auto **GetPublicNym** (const UnallocatedCString &key) const noexcept -> void=0
- virtual auto **GetServerContract** (const UnallocatedCString &key) const noexcept -> void=0
- virtual auto **GetUnitDefinition** (const UnallocatedCString &key) const noexcept -> void=0
- virtual auto **Insert** (const UnallocatedCString &key, const UnallocatedCString &value) const noexcept -> void=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Dht &=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Dht &=0

6.38.1 Detailed Description

The api::network::Dht API is used for accessing functions specific to using a DHT. For example, to query the DHT for a Nym's credentials, or for a server contract, or an asset contract.

The documentation for this class was generated from the following file:

- include/opentxs/api/network/Dht.hpp

6.39 opentxs::api::crypto::Encode Class Reference

`#include <Encode.hpp>`

Public Member Functions

- virtual auto **DataEncode** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual auto **DataEncode** (const Data &input) const -> UnallocatedCString=0
- virtual auto **DataDecode** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual auto **IdentifierEncode** (const Data &input) const -> UnallocatedCString=0
- virtual auto **IdentifierDecode** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual OPENTXS_NO_EXPORT auto **InternalEncode** () const noexcept -> const internal::Encode &=0
- virtual auto **IsBase62** (const UnallocatedCString &str) const -> bool=0
- virtual auto **Nonce** (const std::uint32_t size) const -> OTString=0
- virtual auto **Nonce** (const std::uint32_t size, Data &rawOutput) const -> OTString=0
- virtual auto **RandomFilename** () const -> UnallocatedCString=0
- virtual auto **SanatizeBase58** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual auto **SanatizeBase64** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual auto **Z85Encode** (const Data &input) const -> UnallocatedCString=0
- virtual auto **Z85Encode** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual auto **Z85Decode** (const Data &input) const -> OTData=0
- virtual auto **Z85Decode** (const UnallocatedCString &input) const -> UnallocatedCString=0
- virtual OPENTXS_NO_EXPORT auto **InternalEncode** () noexcept -> internal::Encode &=0

6.39.1 Detailed Description

The api::crypto::encode API is used for encoding and decoding data, and related functions.

The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Encode.hpp

6.40 opentxs::api::session::Endpoints Class Reference

Public Member Functions

- virtual auto AccountUpdate () const noexcept -> std::string_view=0
- virtual auto BlockchainAccountCreated () const noexcept -> std::string_view=0
- virtual auto BlockchainBalance () const noexcept -> std::string_view=0
- virtual auto BlockchainBlockAvailable () const noexcept -> std::string_view=0
- virtual auto BlockchainBlockDownloadQueue () const noexcept -> std::string_view=0
- virtual auto BlockchainMempool () const noexcept -> std::string_view=0
- virtual auto BlockchainNewFilter () const noexcept -> std::string_view=0
- virtual auto BlockchainPeer () const noexcept -> std::string_view=0
- virtual auto BlockchainPeerConnection () const noexcept -> std::string_view=0
- virtual auto BlockchainReorg () const noexcept -> std::string_view=0
- virtual auto BlockchainScanProgress () const noexcept -> std::string_view=0
- virtual auto BlockchainStateChange () const noexcept -> std::string_view=0
- virtual auto BlockchainSyncProgress () const noexcept -> std::string_view=0
- virtual auto BlockchainSyncServerUpdated () const noexcept -> std::string_view=0
- virtual auto BlockchainTransactions () const noexcept -> std::string_view=0
- virtual auto BlockchainTransactions (const identifier::Nym &nym) const noexcept -> std::string_view=0
- virtual auto BlockchainWalletUpdated () const noexcept -> std::string_view=0
- virtual auto ConnectionStatus () const noexcept -> std::string_view=0
- virtual auto ContactUpdate () const noexcept -> std::string_view=0
- virtual auto DhtRequestNym () const noexcept -> std::string_view=0
- virtual auto DhtRequestServer () const noexcept -> std::string_view=0
- virtual auto DhtRequestUnit () const noexcept -> std::string_view=0
- virtual auto FindNym () const noexcept -> std::string_view=0
- virtual auto FindServer () const noexcept -> std::string_view=0
- virtual auto FindUnitDefinition () const noexcept -> std::string_view=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const session::internal::Endpoints &=0
- virtual auto IssuerUpdate () const noexcept -> std::string_view=0
- virtual auto Messagability () const noexcept -> std::string_view=0
- virtual auto MessageLoaded () const noexcept -> std::string_view=0
- virtual auto NymCreated () const noexcept -> std::string_view=0
- virtual auto NymDownload () const noexcept -> std::string_view=0
- virtual auto PairEvent () const noexcept -> std::string_view=0
- virtual auto PeerReplyUpdate () const noexcept -> std::string_view=0
- virtual auto PeerRequestUpdate () const noexcept -> std::string_view=0
- virtual auto PendingBailment () const noexcept -> std::string_view=0
- virtual auto SeedUpdated () const noexcept -> std::string_view=0
- virtual auto ServerReplyReceived () const noexcept -> std::string_view=0
- virtual auto ServerRequestSent () const noexcept -> std::string_view=0
- virtual auto ServerUpdate () const noexcept -> std::string_view=0
- virtual auto Shutdown () const noexcept -> std::string_view=0
- virtual auto TaskComplete () const noexcept -> std::string_view=0
- virtual auto ThreadUpdate (const std::string_view thread) const noexcept -> std::string_view=0
- virtual auto UnitUpdate () const noexcept -> std::string_view=0
- virtual auto WidgetUpdate () const noexcept -> std::string_view=0
- virtual auto WorkflowAccountUpdate () const noexcept -> std::string_view=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> session::internal::Endpoints &=0

6.40.1 Member Function Documentation

6.40.1.1 AccountUpdate()

```
virtual auto opentxs::api::session::Endpoints::AccountUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Account balance update notifications

A subscribe socket can connect to this endpoint to receive AccountUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.2 BlockchainAccountCreated()

```
virtual auto opentxs::api::session::Endpoints::BlockchainAccountCreated ( ) const -> std↩
::string_view  [pure virtual], [noexcept]
```

Blockchain account creation notification

A subscribe socket can connect to this endpoint to receive BlockchainAccountCreated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.3 BlockchainBalance()

```
virtual auto opentxs::api::session::Endpoints::BlockchainBalance ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Blockchain balance notifications

A dealer socket can connect to this endpoint to send and receive BlockchainBalance tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.4 BlockchainBlockAvailable()

```
virtual auto opentxs::api::session::Endpoints::BlockchainBlockAvailable ( ) const -> std↩
::string_view  [pure virtual], [noexcept]
```

Blockchain block available notifications

A subscribe socket can connect to this endpoint to receive BlockchainBlockAvailable tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.5 BlockchainBlockDownloadQueue()

```
virtual auto opentxs::api::session::Endpoints::BlockchainBlockDownloadQueue ( ) const -> std↩
::string_view  [pure virtual], [noexcept]
```

Blockchain block download queue notifications

A subscribe socket can connect to this endpoint to receive BlockchainBlockDownloadQueue tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.6 BlockchainMempool()

```
virtual auto opentxs::api::session::Endpoints::BlockchainMempool ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Blockchain mempool updates

A subscribe socket can connect to this endpoint to receive BlockchainMempoolUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.7 BlockchainNewFilter()

```
virtual auto opentxs::api::session::Endpoints::BlockchainNewFilter ( ) const -> std::string_↩
view  [pure virtual], [noexcept]
```

Blockchain filter oracle notifications

A subscribe socket can connect to this endpoint to receive BlockchainNewFilter tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.8 BlockchainPeer()

```
virtual auto opentxs::api::session::Endpoints::BlockchainPeer ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Blockchain peer connection ready

A subscribe socket can connect to this endpoint to receive BlockchainPeerAdded tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.9 BlockchainPeerConnection()

```
virtual auto opentxs::api::session::Endpoints::BlockchainPeerConnection ( ) const -> std↩
::string_view  [pure virtual], [noexcept]
```

Blockchain peer connection initiated or lost

A subscribe socket can connect to this endpoint to receive BlockchainPeerConnected tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.10 BlockchainReorg()

```
virtual auto opentxs::api::session::Endpoints::BlockchainReorg ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Blockchain reorg and update notifications

A subscribe socket can connect to this endpoint to receive BlockchainNewHeader and BlockchainReorg tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.11 BlockchainScanProgress()

```
virtual auto opentxs::api::session::Endpoints::BlockchainScanProgress ( ) const -> std::string↩
_view  [pure virtual], [noexcept]
```

Blockchain wallet scan progress

A subscribe socket can connect to this endpoint to receive BlockchainWalletScanProgress tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.12 BlockchainStateChange()

```
virtual auto opentxs::api::session::Endpoints::BlockchainStateChange ( ) const -> std::string↩
_view  [pure virtual], [noexcept]
```

Blockchain enabled state change

A subscribe socket can connect to this endpoint to receive BlockchainStateChange tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.13 BlockchainSyncProgress()

```
virtual auto opentxs::api::session::Endpoints::BlockchainSyncProgress ( ) const -> std::string↩
_view  [pure virtual], [noexcept]
```

Blockchain wallet sync progress

A subscribe socket can connect to this endpoint to receive BlockchainSyncProgress tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.14 BlockchainSyncServerUpdated()

```
virtual auto opentxs::api::session::Endpoints::BlockchainSyncServerUpdated ( ) const -> std↩
::string_view  [pure virtual], [noexcept]
```

Blockchain sync server database changes

A subscribe socket can connect to this endpoint to receive SyncServerUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.15 BlockchainTransactions() [1/2]

```
virtual auto opentxs::api::session::Endpoints::BlockchainTransactions ( ) const -> std::string↩
_view  [pure virtual], [noexcept]
```

Blockchain transaction notifications (global)

A subscribe socket can connect to this endpoint to receive BlockchainNewTransaction tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.16 BlockchainTransactions() [2/2]

```
virtual auto opentxs::api::session::Endpoints::BlockchainTransactions (
            const identifier::Nym & nym ) const -> std::string_view  [pure virtual], [noexcept]
```

Blockchain transaction notifications (per-nym)

A subscribe socket can connect to this endpoint to receive BlockchainNewTransaction tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.17 BlockchainWalletUpdated()

```
virtual auto opentxs::api::session::Endpoints::BlockchainWalletUpdated ( ) const -> std↩
::string_view  [pure virtual], [noexcept]
```

Blockchain wallet balance updates

A subscribe socket can connect to this endpoint to receive BlockchainWalletUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.18 ConnectionStatus()

```
virtual auto opentxs::api::session::Endpoints::ConnectionStatus ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Connection state notifications

A subscribe socket can connect to this endpoint to receive OTXConnectionStatus tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.19 ContactUpdate()

```
virtual auto opentxs::api::session::Endpoints::ContactUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Contact account creation notification

A subscribe socket can connect to this endpoint to receive ContactUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.20 DhtRequestNym()

```
virtual auto opentxs::api::session::Endpoints::DhtRequestNym ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Search for a nym in the DHT

A dealer socket can connect to this endpoint to send and receive DHTRequestNym tagged messages.

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.21 DhtRequestServer()

```
virtual auto opentxs::api::session::Endpoints::DhtRequestServer ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Search for a notary in the DHT

A dealer socket can connect to this endpoint to send and receive DHTRequestServer tagged messages.

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.22 DhtRequestUnit()

```
virtual auto opentxs::api::session::Endpoints::DhtRequestUnit ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Search for a unit definition in the DHT

A dealer socket can connect to this endpoint to send and receive DHTRequestUnit tagged messages.

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.23 FindNym()

```
virtual auto opentxs::api::session::Endpoints::FindNym ( ) const -> std::string_view  [pure
virtual], [noexcept]
```

Search for a nym on known notaries

A push socket can connect to this endpoint to send OTXSearchNym tagged messages.

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.24 FindServer()

```
virtual auto opentxs::api::session::Endpoints::FindServer ( ) const -> std::string_view  [pure
virtual], [noexcept]
```

Search for a notary contract on known notaries

A push socket can connect to this endpoint to send OTXSearchServer tagged messages.

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.25 FindUnitDefinition()

```
virtual auto opentxs::api::session::Endpoints::FindUnitDefinition ( ) const -> std::string_↩
view  [pure virtual], [noexcept]
```

Search for a unit definition on known notaries

A push socket can connect to this endpoint to send OTXSearchUnit tagged messages.

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.26 IssuerUpdate()

```
virtual auto opentxs::api::session::Endpoints::IssuerUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Issuer update notifications

A subscribe socket can connect to this endpoint to receive IssuerUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.27 Messagability()

```
virtual auto opentxs::api::session::Endpoints::Messagability ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Contact messagability status

A subscribe socket can connect to this endpoint to receive OTXMessagability tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.28 MessageLoaded()

```
virtual auto opentxs::api::session::Endpoints::MessageLoaded ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Message loaded

A subscribe socket can connect to this endpoint to receive MessageLoaded tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.29 NymCreated()

```
virtual auto opentxs::api::session::Endpoints::NymCreated ( ) const -> std::string_view [pure
virtual], [noexcept]
```

Nym created notifications

A subscribe socket can connect to this endpoint to receive NymCreated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.30 NymDownload()

```
virtual auto opentxs::api::session::Endpoints::NymDownload ( ) const -> std::string_view [pure
virtual], [noexcept]
```

Nym update notifications

A subscribe socket can connect to this endpoint to receive NymUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.31 PairEvent()

```
virtual auto opentxs::api::session::Endpoints::PairEvent ( ) const -> std::string_view [pure
virtual], [noexcept]
```

Node pairing event notification

A subscribe socket can connect to this endpoint to be notified when any peer message related to node pairing is received.

Messages bodies consist of one frame.

- The frame contains a serialized proto::PairEvent message

This endpoint is active for client sessions only.

6.40.1.32 PeerReplyUpdate()

```
virtual auto opentxs::api::session::Endpoints::PeerReplyUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Peer reply event notification

A subscribe socket can connect to this endpoint to be notified when any peer reply is received.

Messages bodies consist of two frame.

- The first frame contains the recipient nym as a serialized string
- The second frame contains a serialized proto::PeerReply message

This endpoint is active for client sessions only.

6.40.1.33 PeerRequestUpdate()

```
virtual auto opentxs::api::session::Endpoints::PeerRequestUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Peer request event notification

A subscribe socket can connect to this endpoint to be notified when any peer request is received.

Messages bodies consist of one frame.

- The first frame contains the recipient nym as a serialized string
- The second frame contains a serialized proto::PeerRequest message

This endpoint is active for client sessions only.

6.40.1.34 PendingBailment()

```
virtual auto opentxs::api::session::Endpoints::PendingBailment ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Pending bailment notification

A subscribe socket can connect to this endpoint to be notified when a pending bailment peer request has been received.

Messages bodies consist of one frame.

- The frame contains a serialized proto::PeerRequest message

This endpoint is active for client sessions only.

6.40.1.35 SeedUpdated()

```
virtual auto opentxs::api::session::Endpoints::SeedUpdated ( ) const -> std::string_view [pure
virtual], [noexcept]
```

HD seed update notifications

A subscribe socket can connect to this endpoint to receive SeedUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.36 ServerReplyReceived()

```
virtual auto opentxs::api::session::Endpoints::ServerReplyReceived ( ) const -> std::string_↩
view  [pure virtual], [noexcept]
```

Server reply notification

A subscribe socket can connect to this endpoint to be notified when any server reply is received.

Messages bodies consist of one frame.

- The frame contains of the message type as a string

This endpoint is active for client sessions only.

6.40.1.37 ServerRequestSent()

```
virtual auto opentxs::api::session::Endpoints::ServerRequestSent ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Server request notification

A subscribe socket can connect to this endpoint to be notified when any request message is sent to a notary.

Messages bodies consist of one frame.

- The frame contains of the message type as a string

This endpoint is active for client sessions only.

6.40.1.38 ServerUpdate()

```
virtual auto opentxs::api::session::Endpoints::ServerUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Server contract update notifications

A subscribe socket can connect to this endpoint to receive NotaryUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.39 Shutdown()

```
virtual auto opentxs::api::session::Endpoints::Shutdown ( ) const -> std::string_view  [pure
virtual], [noexcept]
```

Notification of context shutdown

A subscribe socket can connect to this endpoint to receive Shutdown tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.40 TaskComplete()

```
virtual auto opentxs::api::session::Endpoints::TaskComplete ( ) const -> std::string_view
[pure virtual], [noexcept]
```

Background task completion notification

A subscribe socket can connect to this endpoint to receive OTXTaskComplete tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.41 ThreadUpdate()

```
virtual auto opentxs::api::session::Endpoints::ThreadUpdate (
            const std::string_view thread ) const -> std::string_view  [pure virtual], [noexcept]
```

Activity thread update notification

A subscribe socket can connect to this endpoint to receive ActivityThreadUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.42 UnitUpdate()

```
virtual auto opentxs::api::session::Endpoints::UnitUpdate ( ) const -> std::string_view  [pure
virtual], [noexcept]
```

Unit definition contract update notifications

A subscribe socket can connect to this endpoint to receive UnitDefinitionUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for all session types.

6.40.1.43 WidgetUpdate()

```
virtual auto opentxs::api::session::Endpoints::WidgetUpdate ( ) const -> std::string_view
[pure virtual], [noexcept]
```

UI widget update notification

A subscribe socket can connect to this endpoint to receive UIModelUpdated tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

6.40.1.44 WorkflowAccountUpdate()

```
virtual auto opentxs::api::session::Endpoints::WorkflowAccountUpdate ( ) const -> std::string↩
_view  [pure virtual], [noexcept]
```

Account update notification

A subscribe socket can connect to this endpoint to receive WorkflowAccountUpdate tagged messages

See opentxs/util/WorkTypes.hpp for message format documentation

This endpoint is active for client sessions only.

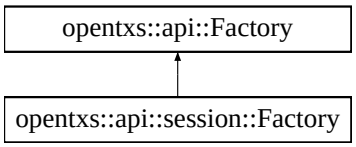
The documentation for this class was generated from the following file:

- include/opentxs/api/session/Endpoints.hpp

6.41 opentxs::api::Factory Class Reference

```
#include <Factory.hpp>
```

Inheritance diagram for opentxs::api::Factory:



Public Member Functions

- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Factory &=0
- virtual auto **Secret** (const std::size_t bytes) const noexcept -> OTSecret=0
- virtual auto **SecretFromBytes** (const ReadView bytes) const noexcept -> OTSecret=0
- virtual auto **SecretFromText** (const std::string_view text) const noexcept -> OTSecret=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Factory &=0

6.41.1 Detailed Description

The top-level Factory API, used for instantiating secrets. A Secret is a piece of data, similar to a string or byte vector. But secrets, unlike normal strings or byte vectors, have additional secrecy requirements. They are used to store, for example, private keys. They have additional security requirements such as wiping their memory to zero when destructed.

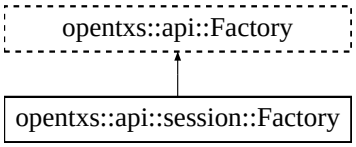
The documentation for this class was generated from the following file:

- include/opentxs/api/Factory.hpp

6.42 opentxs::api::session::Factory Class Reference

`#include <Factory.hpp>`

Inheritance diagram for opentxs::api::session::Factory:



Public Member Functions

- virtual auto **Armored** () const -> OTArmored=0
- virtual auto **Armored** (const UnallocatedCString &input) const -> OTArmored=0
- virtual auto **Armored** (const opentxs::Data &input) const -> OTArmored=0
- virtual auto **Armored** (const opentxs::String &input) const -> OTArmored=0
- virtual auto **Armored** (const opentxs::crypto::Envelope &input) const -> OTArmored=0
- virtual auto **AsymmetricKey** (const opentxs::crypto::Parameters &params, const opentxs::PasswordPrompt &reason, const opentxs::crypto::key::asymmetric::Role role=opentxs::crypto::key::asymmetric::Role::Sign, const VersionNumber version=opentxs::crypto::key::Asymmetric::DefaultVersion) const -> OTAsymmetric↩ Key=0
- virtual auto **BailmentNotice** (const Nym_p &nym, const identifier::Nym &recipientID, const identifier::↩ UnitDefinition &unitID, const identifier::Notary &serverID, const Identifier &requestID, const Unallocated↩ CString &txid, const Amount &amount, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTBailmentNotice=0
- virtual auto **BailmentReply** (const Nym_p &nym, const identifier::Nym &initiator, const Identifier &request, const identifier::Notary &server, const UnallocatedCString &terms, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTBailmentReply=0
- virtual auto **BailmentRequest** (const Nym_p &nym, const identifier::Nym &recipient, const identifier::↩ UnitDefinition &unit, const identifier::Notary &server, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTBailmentRequest=0
- virtual auto **BailmentRequest** (const Nym_p &nym, const ReadView &view) const noexcept(false) -> OTBailmentRequest=0
- virtual auto **BasketContract** (const Nym_p &nym, const UnallocatedCString &shortname, const UnallocatedCString &terms, const std::uint64_t weight, const UnitType unitOfAccount, const Version↩ Number version, const display::Definition &displayDefinition, const Amount &redemptionIncrement) const noexcept(false) -> OTBasketContract=0
- virtual auto **ConnectionReply** (const Nym_p &nym, const identifier::Nym &initiator, const Identifier &request, const identifier::Notary &server, const bool ack, const UnallocatedCString &url, const UnallocatedCString &login, const UnallocatedCString &password, const UnallocatedCString &key, const opentxs::Password↩ Prompt &reason) const noexcept(false) -> OTConnectionReply=0
- virtual auto **ConnectionRequest** (const Nym_p &nym, const identifier::Nym &recipient, const contract↩ ::peer::ConnectionInfoType type, const identifier::Notary &server, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTConnectionRequest=0
- virtual auto **CurrencyContract** (const Nym_p &nym, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const VersionNumber version, const opentxs::↩ PasswordPrompt &reason, const display::Definition &displayDefinition, const Amount &redemptionIncrement) const noexcept(false) -> OTCurrencyContract=0
- virtual auto **Data** () const -> OTData=0
- virtual auto **Data** (const opentxs::Armored &input) const -> OTData=0
- virtual auto **Data** (const opentxs::network::zeromq::Frame &input) const -> OTData=0
- virtual auto **Data** (const std::uint8_t input) const -> OTData=0

- virtual auto **Data** (const std::uint32_t input) const -> OTData=0
- virtual auto **Data** (const UnallocatedVector< unsigned char > &input) const -> OTData=0
- virtual auto **Data** (const UnallocatedVector< std::byte > &input) const -> OTData=0
- virtual auto **DataFromBytes** (const ReadView input) const -> OTData=0
- virtual auto **DataFromHex** (const ReadView input) const -> OTData=0
- virtual auto **Envelope** () const noexcept -> OTEnvelope=0
- virtual auto **Envelope** (const opentxs::Armored &ciphertext) const noexcept(false) -> OTEnvelope=0
- virtual auto **Envelope** (const opentxs::crypto::Envelope::SerializedType &serialized) const noexcept(false) -> OTEnvelope=0
- virtual auto **Envelope** (const opentxs::ReadView &serialized) const noexcept(false) -> OTEnvelope=0
- virtual auto **Identifier** () const -> OTIdentifier=0
- virtual auto **Identifier** (const UnallocatedCString &serialized) const -> OTIdentifier=0
- virtual auto **Identifier** (const opentxs::String &serialized) const -> OTIdentifier=0
- virtual auto **Identifier** (const opentxs::Contract &contract) const -> OTIdentifier=0
- virtual auto **Identifier** (const opentxs::Item &item) const -> OTIdentifier=0
- virtual auto **Identifier** (const ReadView bytes) const -> OTIdentifier=0
- virtual auto **Identifier** (const opentxs::network::zeromq::Frame &bytes) const -> OTIdentifier=0
- virtual OPENTXS_NO_EXPORT auto **InternalSession** () const noexcept -> const internal::Factory &=0
- virtual auto **Keypair** (const opentxs::crypto::Parameters &nymParameters, const VersionNumber version, const opentxs::crypto::key::asymmetric::Role role, const opentxs::PasswordPrompt &reason) const -> OTKeypair=0
- virtual auto **Keypair** (const UnallocatedCString &fingerprint, const Bip32Index nym, const Bip32Index credset, const Bip32Index credindex, const opentxs::crypto::EcdsaCurve &curve, const opentxs::crypto::key↩ ::asymmetric::Role role, const opentxs::PasswordPrompt &reason) const -> OTKeypair=0
- virtual auto **Mint** () const noexcept -> otx::blind::Mint=0
- virtual auto **Mint** (const otx::blind::CashType type) const noexcept -> otx::blind::Mint=0
- virtual auto **Mint** (const identifier::Notary &notary, const identifier::UnitDefinition &unit) const noexcept -> otx::blind::Mint=0
- virtual auto **Mint** (const otx::blind::CashType type, const identifier::Notary &notary, const identifier::Unit↩ Definition &unit) const noexcept -> otx::blind::Mint=0
- virtual auto **Mint** (const identifier::Notary &notary, const identifier::Nym &serverNym, const identifier::Unit↩ Definition &unit) const noexcept -> otx::blind::Mint=0
- virtual auto **Mint** (const otx::blind::CashType type, const identifier::Notary &notary, const identifier::Nym &serverNym, const identifier::UnitDefinition &unit) const noexcept -> otx::blind::Mint=0
- virtual auto **NymID** () const -> OTNymID=0
- virtual auto **NymID** (const UnallocatedCString &serialized) const -> OTNymID=0
- virtual auto **NymID** (const opentxs::String &serialized) const -> OTNymID=0
- virtual auto **NymID** (const opentxs::network::zeromq::Frame &bytes) const -> OTNymID=0
- virtual auto **NymIDFromPaymentCode** (const UnallocatedCString &serialized) const -> OTNymID=0
- virtual auto **OutbailmentReply** (const Nym_p &nym, const identifier::Nym &initiator, const opentxs::Identifier &request, const identifier::Notary &server, const UnallocatedCString &terms, const opentxs::Password↩ Prompt &reason) const noexcept(false) -> OTOutbailmentReply=0
- virtual auto **OutbailmentRequest** (const Nym_p &nym, const identifier::Nym &recipientID, const identifier↩ ::UnitDefinition &unitID, const identifier::Notary &serverID, const Amount &amount, const UnallocatedCString &terms, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTOutbailmentRequest=0
- virtual auto **PasswordPrompt** (const UnallocatedCString &text) const -> OTPasswordPrompt=0
- virtual auto **PasswordPrompt** (const opentxs::PasswordPrompt &rhs) const -> OTPasswordPrompt=0
- virtual auto **PaymentCode** (const UnallocatedCString &base58) const noexcept -> opentxs::Payment↩ Code=0
- virtual auto **PaymentCode** (const ReadView &serialized) const noexcept -> opentxs::PaymentCode=0
- virtual auto **PaymentCode** (const UnallocatedCString &seed, const Bip32Index nym, const std::uint8_t version, const opentxs::PasswordPrompt &reason, const bool bitmessage=false, const std::uint8_t bitmessage↩ Version=0, const std::uint8_t bitmessageStream=0) const noexcept -> opentxs::PaymentCode=0
- virtual auto **PeerObject** (const Nym_p &senderNym, const UnallocatedCString &message) const -> std↩ ::unique_ptr< opentxs::PeerObject >=0

- virtual auto **PeerObject** (const Nym_p &senderNym, const UnallocatedCString &payment, const bool isPayment) const -> std::unique_ptr< opentxs::PeerObject >=0
- virtual auto **PeerObject** (const Nym_p &senderNym, otx::blind::Purse &&purse) const -> std::unique_ptr< opentxs::PeerObject >=0
- virtual auto **PeerObject** (const OTPeerRequest request, const OTPeerReply reply, const VersionNumber version) const -> std::unique_ptr< opentxs::PeerObject >=0
- virtual auto **PeerObject** (const OTPeerRequest request, const VersionNumber version) const -> std::unique_ptr< opentxs::PeerObject >=0
- virtual auto **PeerObject** (const Nym_p &recipientNym, const opentxs::Armored &encrypted, const opentxs::PasswordPrompt &reason) const -> std::unique_ptr< opentxs::PeerObject >=0
- virtual auto **PeerReply** () const noexcept -> OTPeerReply=0
- virtual auto **PeerReply** (const Nym_p &nym, const ReadView &view) const noexcept(false) -> OTPeerReply=0
- virtual auto **PeerRequest** () const noexcept -> OTPeerRequest=0
- virtual auto **PeerRequest** (const Nym_p &nym, const ReadView &view) const noexcept(false) -> OTPeerRequest=0
- virtual auto **Purse** (const otx::context::Server &context, const identifier::UnitDefinition &unit, const otx::blind::Mint &mint, const Amount &totalValue, const opentxs::PasswordPrompt &reason) const noexcept -> otx::blind::Purse=0
- virtual auto **Purse** (const otx::context::Server &context, const identifier::UnitDefinition &unit, const otx::blind::Mint &mint, const Amount &totalValue, const otx::blind::CashType type, const opentxs::PasswordPrompt &reason) const noexcept -> otx::blind::Purse=0
- virtual auto **Purse** (const identity::Nym &owner, const identifier::Notary &server, const identifier::UnitDefinition &unit, const opentxs::PasswordPrompt &reason) const noexcept -> otx::blind::Purse=0
- virtual auto **Purse** (const identity::Nym &owner, const identifier::Notary &server, const identifier::UnitDefinition &unit, const otx::blind::CashType type, const opentxs::PasswordPrompt &reason) const noexcept -> otx::blind::Purse=0
- virtual auto **ReplyAcknowledgement** (const Nym_p &nym, const identifier::Nym &initiator, const opentxs::Identifier &request, const identifier::Notary &server, const contract::peer::PeerRequestType type, const bool &ack, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTReplyAcknowledgement=0
- virtual auto **SecurityContract** (const Nym_p &nym, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const VersionNumber version, const opentxs::PasswordPrompt &reason, const display::Definition &displayDefinition, const Amount &redemptionIncrement) const noexcept(false) -> OTSecurityContract=0
- virtual auto **ServerContract** () const noexcept(false) -> OTServerContract=0
- virtual auto **ServerID** () const -> OTNotaryID=0
- virtual auto **ServerID** (const UnallocatedCString &serialized) const -> OTNotaryID=0
- virtual auto **ServerID** (const opentxs::String &serialized) const -> OTNotaryID=0
- virtual auto **ServerID** (const opentxs::network::zeromq::Frame &bytes) const -> OTNotaryID=0
- virtual auto **StoreSecret** (const Nym_p &nym, const identifier::Nym &recipientID, const contract::peer::SecretType type, const UnallocatedCString &primary, const UnallocatedCString &secondary, const identifier::Notary &server, const opentxs::PasswordPrompt &reason) const noexcept(false) -> OTStoreSecret=0
- virtual auto SymmetricKey () const -> OTSymmetricKey=0
- virtual auto SymmetricKey (const opentxs::crypto::SymmetricProvider &engine, const opentxs::PasswordPrompt &password, const opentxs::crypto::key::symmetric::Algorithm mode=opentxs::crypto::key::symmetric::Algorithm::Error) const -> OTSymmetricKey=0
- virtual auto SymmetricKey (const opentxs::crypto::SymmetricProvider &engine, const opentxs::Secret &seed, const std::uint64_t operations, const std::uint64_t difficulty, const std::size_t size, const opentxs::crypto::key::symmetric::Source type) const -> OTSymmetricKey=0
- virtual auto **SymmetricKey** (const opentxs::crypto::SymmetricProvider &engine, const opentxs::Secret &seed, const ReadView salt, const std::uint64_t operations, const std::uint64_t difficulty, const std::uint64_t parallel, const std::size_t size, const opentxs::crypto::key::symmetric::Source type) const -> OTSymmetricKey=0
- virtual auto SymmetricKey (const opentxs::crypto::SymmetricProvider &engine, const opentxs::Secret &raw, const opentxs::PasswordPrompt &reason) const -> OTSymmetricKey=0

- virtual auto **UnitID** () const -> OTUnitID=0
- virtual auto **UnitID** (const UnallocatedCString &serialized) const -> OTUnitID=0
- virtual auto **UnitID** (const opentxs::String &serialized) const -> OTUnitID=0
- virtual auto **UnitID** (const opentxs::network::zeromq::Frame &bytes) const -> OTUnitID=0
- virtual auto **UnitDefinition** () const noexcept -> OTUnitDefinition=0
- virtual OPENTXS_NO_EXPORT auto **InternalSession** () noexcept -> internal::Factory &=0

6.42.1 Detailed Description

The Factory API for opentxs sessions, used for instantiating many different object types native to opentxs.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 ~Factory()

```
OPENTXS_NO_EXPORT opentxs::api::session::Factory::~Factory ( )  [override], [virtual], [default]
```

Reimplemented from opentxs::api::Factory.

6.42.3 Member Function Documentation

6.42.3.1 SymmetricKey() [1/4]

```
virtual auto opentxs::api::session::Factory::SymmetricKey ( ) const -> OTSymmetricKey  [pure
virtual]
```

Generate a blank, invalid key

6.42.3.2 SymmetricKey() [2/4]

```
virtual auto opentxs::api::session::Factory::SymmetricKey (
            const opentxs::crypto::SymmetricProvider & engine,
            const opentxs::PasswordPrompt & password,
            const opentxs::crypto::key::symmetric::Algorithm mode = opentxs::crypto::key:↩
:symmetric::Algorithm::Error ) const -> OTSymmetricKey  [pure virtual]
```

Derive a new, random symmetric key

Parameters

in	*engine*	A reference to the crypto library to be bound to the instance
in	*password*	Optional key password information.
in	*mode*	The symmetric algorithm for which to generate an appropriate key

6.42.3.3 SymmetricKey() [3/4]

```
virtual auto opentxs::api::session::Factory::SymmetricKey (
            const opentxs::crypto::SymmetricProvider & engine,
            const opentxs::Secret & raw,
            const opentxs::PasswordPrompt & reason ) const -> OTSymmetricKey  [pure virtual]
```

Construct a symmetric key from an existing Secret

Parameters

in	*engine*	A reference to the crypto library to be bound to the instance
in	*raw*	An existing, unencrypted binary or text secret

6.42.3.4 SymmetricKey() [4/4]

```
virtual auto opentxs::api::session::Factory::SymmetricKey (
            const opentxs::crypto::SymmetricProvider & engine,
            const opentxs::Secret & seed,
            const std::uint64_t operations,
            const std::uint64_t difficulty,
            const std::size_t size,
            const opentxs::crypto::key::symmetric::Source type ) const -> OTSymmetricKey
[pure virtual]
```

Derive a symmetric key from a seed

Parameters

in	*seed*	A binary or text seed to be expanded into a secret key
in	*salt*	
in	*operations*	The number of iterations/operations the KDF should perform
in	*difficulty*	A type-specific difficulty parameter used by the KDF.
in	*size*	The target number of bytes for the derived secret key
in	*type*	The KDF to be used for the derivation process

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Factory.hpp

6.43 opentxs::api::crypto::Hash Class Reference

```
#include <Hash.hpp>
```

Public Member Functions

- virtual auto **Digest** (const opentxs::crypto::HashType hashType, const ReadView data, const AllocateOutput destination) const noexcept -> bool=0
- virtual auto **Digest** (const opentxs::crypto::HashType hashType, const opentxs::network::zeromq::Frame &data, const AllocateOutput destination) const noexcept -> bool=0
- virtual auto **Digest** (const std::uint32_t type, const ReadView data, const AllocateOutput encoded↩ Destination) const noexcept -> bool=0
- virtual auto **HMAC** (const opentxs::crypto::HashType hashType, const ReadView key, const ReadView &data, const AllocateOutput digest) const noexcept -> bool=0
- virtual OPENTXS_NO_EXPORT auto **InternalHash** () const noexcept -> const internal::Hash &=0
- virtual auto **MurmurHash3_32** (const std::uint32_t &key, const Data &data, std::uint32_t &output) const noexcept -> void=0
- virtual auto **PKCS5_PBKDF2_HMAC** (const Data &input, const Data &salt, const std::size_t iterations, const opentxs::crypto::HashType hashType, const std::size_t bytes, Data &output) const noexcept -> bool=0
- virtual auto **PKCS5_PBKDF2_HMAC** (const Secret &input, const Data &salt, const std::size_t iterations, const opentxs::crypto::HashType hashType, const std::size_t bytes, Data &output) const noexcept -> bool=0
- virtual auto **PKCS5_PBKDF2_HMAC** (const UnallocatedCString &input, const Data &salt, const std::size_t iterations, const opentxs::crypto::HashType hashType, const std::size_t bytes, Data &output) const noexcept -> bool=0
- virtual auto **Scrypt** (const ReadView input, const ReadView salt, const std::uint64_t N, const std::uint32_t r, const std::uint32_t p, const std::size_t bytes, AllocateOutput writer) const noexcept -> bool=0
- virtual OPENTXS_NO_EXPORT auto **InternalHash** () noexcept -> internal::Hash &=0

6.43.1 Detailed Description

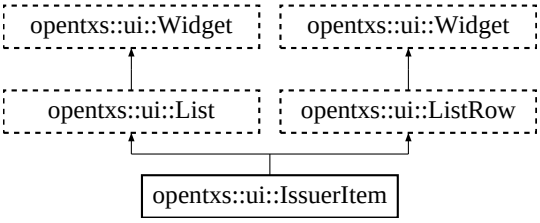
The api::crypto::Hash API contains various hash-related functions.

The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Hash.hpp

6.44 opentxs::ui::IssuerItem Class Reference

Inheritance diagram for opentxs::ui::IssuerItem:



Public Member Functions

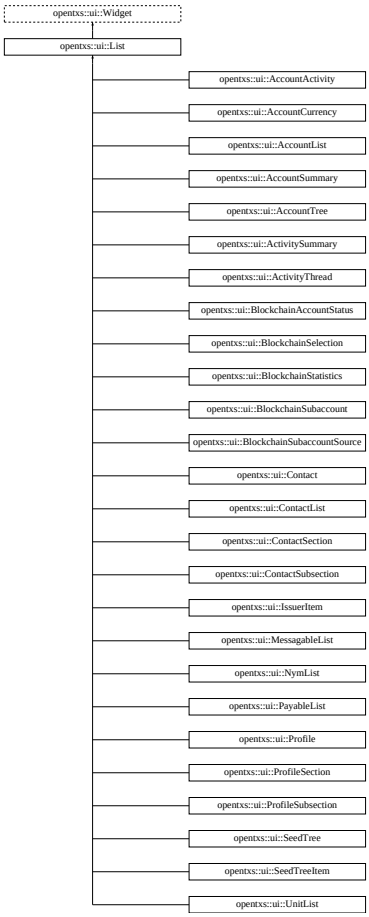
- virtual auto **ConnectionState** () const noexcept -> bool=0
- virtual auto **Debug** () const noexcept -> UnallocatedCString=0
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::AccountSummaryItem >=0
- virtual auto **Name** () const noexcept -> UnallocatedCString=0
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::AccountSummaryItem >=0
- virtual auto **Trusted** () const noexcept -> bool=0

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/IssuerItem.hpp

6.45 opentxs::ui::List Class Reference

Inheritance diagram for opentxs::ui::List:



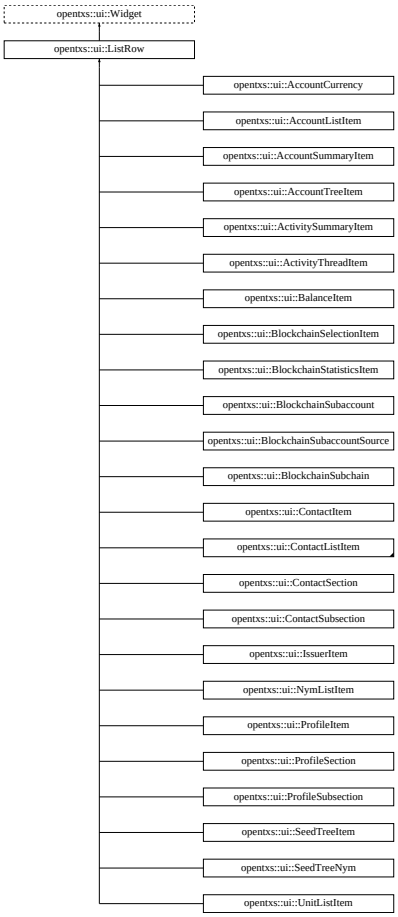
Additional Inherited Members

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/List.hpp

6.46 opentxs::ui::ListRow Class Reference

Inheritance diagram for opentxs::ui::ListRow:



Public Member Functions

- virtual auto **Last** () const noexcept -> bool=0
- virtual auto **Valid** () const noexcept -> bool=0

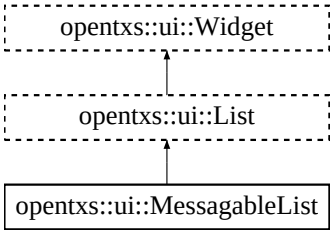
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ListRow.hpp

6.47 opentxs::ui::MessagableList Class Reference

`#include <MessagableList.hpp>`

Inheritance diagram for opentxs::ui::MessagableList:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactListItem >=0
  returns the first row, containing a valid ContactListItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ContactListItem >=0
  returns the next row, containing a valid ContactListItem or an empty smart pointer (if at end of list).

6.47.1 Detailed Description

Like ContactList, this model manages a set of rows containing the Contacts in the wallet. However, this model only contains contacts that are messagable. Like ContactList, each row is a ContactListItem.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/MessagableList.hpp

6.48 opentxs::api::network::Network Class Reference

`#include <Network.hpp>`

Public Member Functions

- auto **Asio** () const noexcept -> const network::Asio &
- auto **Blockchain** () const noexcept -> const network::Blockchain &
- auto **DHT** () const noexcept -> const network::Dht &
- auto **ZeroMQ** () const noexcept -> const opentxs::network::zeromq::Context &
- OPENTXS_NO_EXPORT auto **Shutdown** () noexcept -> void
- OPENTXS_NO_EXPORT **Network** (Imp *) noexcept

6.48.1 Detailed Description

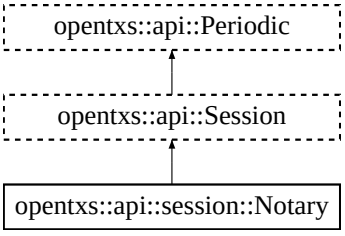
The top-level network API. Used for accessing the Asio API, the Blockchain network API, the DHT network API, and the ZeroMQ network API.

The documentation for this class was generated from the following file:

- include/opentxs/api/network/Network.hpp

6.49 opentxs::api::session::Notary Class Reference

Inheritance diagram for opentxs::api::session::Notary:



Public Member Functions

- virtual auto DropIncoming (const int count) const -> void=0
- virtual auto DropOutgoing (const int count) const -> void=0
- virtual auto **GetAdminNym** () const -> UnallocatedCString=0
- virtual auto **GetAdminPassword** () const -> UnallocatedCString=0
- virtual auto **GetPrivateMint** (const identifier::UnitDefinition &unitid, std::uint32_t series) const noexcept -> otx::blind::Mint &=0
- virtual auto **GetPublicMint** (const identifier::UnitDefinition &unitID) const noexcept -> otx::blind::Mint &=0
- virtual auto **GetUserName** () const -> UnallocatedCString=0
- virtual auto **GetUserTerms** () const -> UnallocatedCString=0
- virtual auto **ID** () const -> const identifier::Notary &=0
- virtual OPENTXS_NO_EXPORT auto **InternalNotary** () const noexcept -> const session::internal::Notary &=0
- virtual auto **NymID** () const -> const identifier::Nym &=0
- virtual auto **ScanMints** () const -> void=0
- virtual auto **Server** () const -> opentxs::server::Server &=0
- virtual auto **SetMintKeySize** (const std::size_t size) const -> void=0
- virtual auto **UpdateMint** (const identifier::UnitDefinition &unitID) const -> void=0
- virtual OPENTXS_NO_EXPORT auto **InternalNotary** () noexcept -> session::internal::Notary &=0

Static Public Member Functions

- static auto **DefaultMintKeyBytes** () noexcept -> std::size_t

6.49.1 Member Function Documentation

6.49.1.1 DropIncoming()

```
virtual auto opentxs::api::session::Notary::DropIncoming (
            const int count ) const -> void  [pure virtual]
```

Drop a specified number of incoming requests for testing purposes

6.49.1.2 DropOutgoing()

```
virtual auto opentxs::api::session::Notary::DropOutgoing (
            const int count ) const -> void  [pure virtual]
```

Drop a specified number of outgoing replies for testing purposes

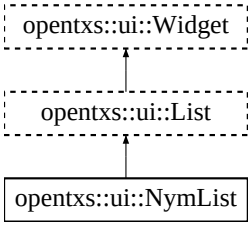
The documentation for this class was generated from the following file:

- include/opentxs/api/session/Notary.hpp

6.50 opentxs::ui::NymList Class Reference

`#include <NymList.hpp>`

Inheritance diagram for opentxs::ui::NymList:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::NymListItem >=0
  returns the first row, containing a valid NymListItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::NymListItem >=0
  returns the next row, containing a valid NymListItem or an empty smart pointer (if at end of list).

6.50.1 Detailed Description

This model manages a set of rows containing the Nyms (user identities) in the wallet. Each row contains a NymListItem model representing a Nym.

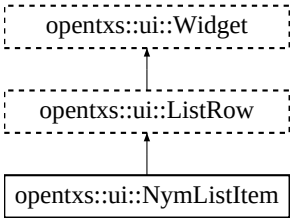
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/NymList.hpp

6.51 opentxs::ui::NymListItem Class Reference

`#include <NymListItem.hpp>`

Inheritance diagram for opentxs::ui::NymListItem:



Public Member Functions

- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns the display name for this Nym.
- virtual auto **NymID** () const noexcept -> UnallocatedCString=0

  Returns the NymID for this Nym.

6.51.1 Detailed Description

This model describes a single Nym from the NymList.

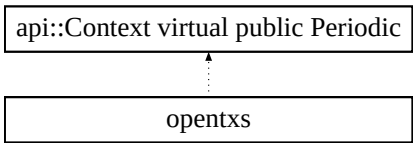
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/NymListItem.hpp

6.52 opentxs Class Reference

`#include <Context.hpp>`

Inheritance diagram for opentxs:



Public Types

- using **ShutdownCallback** = std::function< void()>
- using **PeriodicTask** = std::function< void()>
- using **NymLambda** = std::function< void(const proto::Nym &)>
- using **ServerLambda** = std::function< void(const proto::ServerContract &)>
- using **UnitLambda** = std::function< void(const proto::UnitDefinition &)>
- using AccountInfo = std::tuple< OTIdentifier, OTNymID, OTNotaryID, OTUnitID >
- using **OTUIAccountListItem** = SharedPimpl< ui::AccountListItem >
- using **OTUIAccountSummaryItem** = SharedPimpl< ui::AccountSummaryItem >
- using **OTUIActivitySummaryItem** = SharedPimpl< ui::ActivitySummaryItem >
- using **OTUIActivityThreadItem** = SharedPimpl< ui::ActivityThreadItem >
- using **OTUIBalanceItem** = SharedPimpl< ui::BalanceItem >
- using **OTUIBlockchainSelectionItem** = SharedPimpl< ui::BlockchainSelectionItem >
- using **OTUIBlockchainStatisticsItem** = SharedPimpl< ui::BlockchainStatisticsItem >
- using **OTUIContactItem** = SharedPimpl< ui::ContactItem >
- using **OTUIContactListItem** = SharedPimpl< ui::ContactListItem >
- using **OTUIContactSection** = SharedPimpl< ui::ContactSection >
- using **OTUIContactSubsection** = SharedPimpl< ui::ContactSubsection >
- using **OTUIIssuerItem** = SharedPimpl< ui::IssuerItem >
- using **OTUIPayableListItem** = SharedPimpl< ui::PayableListItem >
- using **OTUIProfileItem** = SharedPimpl< ui::ProfileItem >
- using **OTUIProfileSection** = SharedPimpl< ui::ProfileSection >
- using **OTUIProfileSubsection** = SharedPimpl< ui::ProfileSubsection >
- using **OTUIUnitListItem** = SharedPimpl< ui::UnitListItem >

Public Member Functions

- virtual auto **Asio** () const noexcept -> const network::Asio &=0

  Returns a handle to the ASIO API.
- virtual auto ClientSession (const int instance) const noexcept(false) -> const api::session::Client &=0
- virtual auto **ClientSessionCount** () const noexcept -> std::size_t=0

  Returns the number of client sessions.
- virtual auto **Config** (const UnallocatedCString &path) const noexcept -> const api::Settings &=0

  Returns the settings for a given config file.
- virtual auto **Crypto** () const noexcept -> const api::Crypto &=0

  Returns a handle to the top-level crypto API.
- virtual auto **Factory** () const noexcept -> const api::Factory &=0

  Returns a handle to the top-level Factory API.
- virtual auto HandleSignals (ShutdownCallback *callback=nullptr) const noexcept -> void=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Context &=0
- virtual auto NotarySession (const int instance) const noexcept(false) -> const session::Notary &=0
- virtual auto **NotarySessionCount** () const noexcept -> std::size_t=0

  Returns a count of the notary sessions.
- virtual auto **ProfileId** () const noexcept -> UnallocatedCString=0
- virtual OPENTXS_NO_EXPORT auto **QtRootObject** () const noexcept -> QObject *=0
- virtual auto **RPC** (const rpc::request::Base &command) const noexcept -> std::unique_ptr< rpc::response↩ ::Base >=0

  Used for sending RPC requests. Returns RPC response.
- virtual auto **RPC** (const ReadView command, const AllocateOutput response) const noexcept -> bool=0
- virtual auto StartClientSession (const Options &args, const int instance) const -> const api::session::Client &=0
- virtual auto **StartClientSession** (const int instance) const -> const api::session::Client &=0
- virtual auto **StartClientSession** (const Options &args, const int instance, const UnallocatedCString &recoverWords, const UnallocatedCString &recoverPassphrase) const -> const api::session::Client &=0
- virtual auto StartNotarySession (const Options &args, const int instance) const -> const session::Notary &=0
- virtual auto **StartNotarySession** (const int instance) const -> const session::Notary &=0
- virtual auto ZAP () const noexcept -> const api::network::ZAP &=0
- virtual auto **ZMQ** () const noexcept -> const opentxs::network::zeromq::Context &=0

  Returns a handle to the top-level ZMQ API.
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Context &=0

Static Public Member Functions

- static auto PrepareSignalHandling () noexcept -> void
- static auto **SuggestFolder** (const UnallocatedCString &app) noexcept -> UnallocatedCString

Protected Member Functions

- **Context** ()=default

6.52.1 Detailed Description

The top-level Context for the OT API. Child class of Periodic. Both Client and Server contexts are derived from this class.

6.52.2 Member Typedef Documentation

6.52.2.1 AccountInfo

```
using opentxs::AccountInfo = std::tuple<OTIdentifier, OTNymID, OTNotaryID, OTUnitID>
```

AccountInfo: accountID, nymID, serverID, unitID

6.52.3 Member Function Documentation

6.52.3.1 ClientSession()

```
virtual auto opentxs::ClientSession (
            const int instance ) const -> const api::session::Client &  [pure virtual], [noexcept]
```

Throws std::out_of_range if the specified session does not exist.

6.52.3.2 HandleSignals()

```
virtual auto opentxs::HandleSignals (
            ShutdownCallback * callback = nullptr ) const -> void  [pure virtual], [noexcept]
```

WARNING You must call PrepareSignalHandling() prior to initializating the context if you intend to use this function

6.52.3.3 NotarySession()

```
virtual auto opentxs::NotarySession (
            const int instance ) const -> const session::Notary &  [pure virtual], [noexcept]
```

Throws std::out_of_range if the specified session does not exist.

6.52.3.4 PrepareSignalHandling()

```
static auto opentxs::PrepareSignalHandling ( ) -> void  [static], [noexcept]
```

NOTE You must call PrepareSignalHandling() prior to initializating the context if you intend to use signal handling

6.52.3.5 StartClientSession()

```
virtual auto opentxs::StartClientSession (
            const Options & args,
            const int instance ) const -> const api::session::Client &  [pure virtual]
```

Start up a new client session

If the specified instance exists, it will be returned.

Otherwise the next instance will be created

6.52.3.6 StartNotarySession()

```
virtual auto opentxs::StartNotarySession (
            const Options & args,
            const int instance ) const -> const session::Notary &  [pure virtual]
```

Start up a new server session

If the specified instance exists, it will be returned.

Otherwise the next instance will be created

6.52.3.7 ZAP()

```
virtual auto opentxs::ZAP ( ) const -> const api::network::ZAP &  [pure virtual], [noexcept]
```

Access ZAP configuration API

The documentation for this class was generated from the following files:

- include/opentxs/api/Context.hpp
- include/opentxs/api/Periodic.hpp
- include/opentxs/api/session/Storage.hpp
- include/opentxs/api/session/Wallet.hpp
- include/opentxs/interface/ui/AccountListItem.hpp
- include/opentxs/interface/ui/AccountSummaryItem.hpp
- include/opentxs/interface/ui/ActivitySummaryItem.hpp
- include/opentxs/interface/ui/ActivityThreadItem.hpp
- include/opentxs/interface/ui/BalanceItem.hpp
- include/opentxs/interface/ui/BlockchainSelectionItem.hpp
- include/opentxs/interface/ui/BlockchainStatisticsItem.hpp
- include/opentxs/interface/ui/ContactItem.hpp
- include/opentxs/interface/ui/ContactListItem.hpp
- include/opentxs/interface/ui/ContactSection.hpp
- include/opentxs/interface/ui/ContactSubsection.hpp
- include/opentxs/interface/ui/IssuerItem.hpp
- include/opentxs/interface/ui/PayableListItem.hpp
- include/opentxs/interface/ui/ProfileItem.hpp
- include/opentxs/interface/ui/ProfileSection.hpp
- include/opentxs/interface/ui/ProfileSubsection.hpp
- include/opentxs/interface/ui/UnitListItem.hpp

6.53 opentxs::api::session::OTX Class Reference

Public Types

- using **TaskID** = int
- using **MessageID** = OTIdentifier
- using **Result** = std::pair< otx::LastReplyStatus, std::shared_ptr< Message > >
- using **Future** = std::shared_future< Result >
- using **BackgroundTask** = std::pair< TaskID, Future >
- using **Finished** = std::shared_future< void >

Public Member Functions

- virtual auto **AcknowledgeBailment** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID, const Identifier &requestID, const UnallocatedCString &instructions, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **AcknowledgeNotice** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &recipientID, const Identifier &requestID, const bool ack, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **AcknowledgeOutbailment** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &recipientID, const Identifier &requestID, const UnallocatedCString &details, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **AcknowledgeConnection** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &recipientID, const Identifier &requestID, const bool ack, const UnallocatedCString &url, const UnallocatedCString &login, const UnallocatedCString &password, const UnallocatedCString &key, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **AutoProcessInboxEnabled** () const -> bool=0
- virtual auto **CanDeposit** (const identifier::Nym &recipientNymID, const OTPayment &payment) const -> otx::client::Depositability=0
- virtual auto **CanDeposit** (const identifier::Nym &recipientNymID, const Identifier &accountID, const OTPayment &payment) const -> otx::client::Depositability=0
- virtual auto **CanMessage** (const identifier::Nym &senderNymID, const Identifier &recipientContactID, const bool startIntroductionServer=true) const -> otx::client::Messagability=0
- virtual auto **CheckTransactionNumbers** (const identifier::Nym &nym, const identifier::Notary &serverID, const std::size_t quantity) const -> bool=0
- virtual auto **ContextIdle** (const identifier::Nym &nym, const identifier::Notary &server) const -> Finished=0
- virtual auto DepositCheques (const identifier::Nym &nymID) const -> std::size_t=0
- virtual auto DepositCheques (const identifier::Nym &nymID, const UnallocatedSet< OTIdentifier > &chequeIDs) const -> std::size_t=0
- virtual auto **DepositPayment** (const identifier::Nym &recipientNymID, const std::shared_ptr< const OTPayment > &payment) const -> BackgroundTask=0
- virtual auto **DepositPayment** (const identifier::Nym &recipientNymID, const Identifier &accountID, const std::shared_ptr< const OTPayment > &payment) const -> BackgroundTask=0
- virtual void DisableAutoaccept () const =0
- virtual auto **DownloadMint** (const identifier::Nym &nym, const identifier::Notary &server, const identifier::UnitDefinition &unit) const -> BackgroundTask=0
- virtual auto **DownloadNym** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID) const -> BackgroundTask=0
- virtual auto **DownloadNymbox** (const identifier::Nym &localNymID, const identifier::Notary &serverID) const -> BackgroundTask=0
- virtual auto **DownloadServerContract** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Notary &contractID) const -> BackgroundTask=0
- virtual auto **DownloadUnitDefinition** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::UnitDefinition &contractID) const -> BackgroundTask=0

- virtual auto **FindNym** (const identifier::Nym &nymID) const -> BackgroundTask=0
- virtual auto **FindNym** (const identifier::Nym &nymID, const identifier::Notary &serverIDHint) const -> BackgroundTask=0
- virtual auto **FindServer** (const identifier::Notary &serverID) const -> BackgroundTask=0
- virtual auto **FindUnitDefinition** (const identifier::UnitDefinition &unit) const -> BackgroundTask=0
- virtual auto **InitiateBailment** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID, const identifier::UnitDefinition &instrumentDefinitionID, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **InitiateOutbailment** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID, const identifier::UnitDefinition &instrumentDefinitionID, const Amount amount, const UnallocatedCString &message, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **InitiateRequestConnection** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID, const contract::peer::ConnectionInfoType &type, const otx↩ ::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **InitiateStoreSecret** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID, const contract::peer::SecretType &type, const UnallocatedCString &primary, const UnallocatedCString &secondary, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::OTX &=0
- virtual auto **IntroductionServer** () const -> const identifier::Notary &=0
- virtual auto **IssueUnitDefinition** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::UnitDefinition &unitID, const UnitType advertise=UnitType::Error, const UnallocatedCString &label="") const -> BackgroundTask=0
- virtual auto **MessageContact** (const identifier::Nym &senderNymID, const Identifier &contactID, const UnallocatedCString &message, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **MessageStatus** (const TaskID taskID) const -> std::pair< otx::client::ThreadStatus, MessageID >=0
- virtual auto **NotifyBailment** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::Nym &targetNymID, const identifier::UnitDefinition &instrumentDefinitionID, const Identifier &requestID, const UnallocatedCString &txid, const Amount amount, const otx::client::SetID setID={}) const -> BackgroundTask=0
- virtual auto **PayContact** (const identifier::Nym &senderNymID, const Identifier &contactID, std::shared_ptr< const OTPayment > payment) const -> BackgroundTask=0
- virtual auto **PayContactCash** (const identifier::Nym &senderNymID, const Identifier &contactID, const Identifier &workflowID) const -> BackgroundTask=0
- virtual auto **ProcessInbox** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const Identifier &accountID) const -> BackgroundTask=0
- virtual auto **PublishServerContract** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const Identifier &contractID) const -> BackgroundTask=0
- virtual void **Refresh** () const =0
- virtual auto **RefreshCount** () const -> std::uint64_t=0
- virtual auto **RegisterAccount** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const identifier::UnitDefinition &unitID, const UnallocatedCString &label="") const -> BackgroundTask=0
- virtual auto **RegisterNym** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const bool resync=false) const -> BackgroundTask=0
- virtual auto **RegisterNymPublic** (const identifier::Nym &nymID, const identifier::Notary &server, const bool setContactData, const bool forcePrimary=false, const bool resync=false) const -> BackgroundTask=0
- virtual auto **SetIntroductionServer** (const contract::Server &contract) const -> OTNotaryID=0
- virtual auto **SendCheque** (const identifier::Nym &localNymID, const Identifier &sourceAccountID, const Identifier &recipientContactID, const Amount value, const UnallocatedCString &memo, const Time valid↩ From=Clock::now(), const Time validTo=(Clock::now()+std::chrono::hours(OT_CHEQUE_HOURS))) const -> BackgroundTask=0
- virtual auto **SendExternalTransfer** (const identifier::Nym &localNymID, const identifier::Notary &server↩ ID, const Identifier &sourceAccountID, const Identifier &targetAccountID, const Amount &value, const UnallocatedCString &memo) const -> BackgroundTask=0
- virtual auto **SendTransfer** (const identifier::Nym &localNymID, const identifier::Notary &serverID, const Identifier &sourceAccountID, const Identifier &targetAccountID, const Amount &value, const UnallocatedCString &memo) const -> BackgroundTask=0

- virtual void **StartIntroductionServer** (const identifier::Nym &localNymID) const =0
- virtual auto **Status** (const TaskID taskID) const -> otx::client::ThreadStatus=0
- virtual auto **WithdrawCash** (const identifier::Nym &nymID, const identifier::Notary &serverID, const Identifier &account, const Amount value) const -> BackgroundTask=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::OTX &=0

6.53.1 Member Function Documentation

6.53.1.1 DepositCheques() [1/2]

```
virtual auto opentxs::api::session::OTX::DepositCheques (
            const identifier::Nym & nymID ) const -> std::size_t  [pure virtual]
```

Deposit all available cheques for specified nym

Returns

the number of cheques queued for deposit

6.53.1.2 DepositCheques() [2/2]

```
virtual auto opentxs::api::session::OTX::DepositCheques (
            const identifier::Nym & nymID,
            const UnallocatedSet< OTIdentifier > & chequeIDs ) const -> std::size_t  [pure
virtual]
```

Deposit the specified list of cheques for specified nym

If the list of chequeIDs is empty, then all cheques will be deposited

Returns

the number of cheques queued for deposit

6.53.1.3 DisableAutoaccept()

```
virtual void opentxs::api::session::OTX::DisableAutoaccept ( ) const  [pure virtual]
```

Used by unit tests

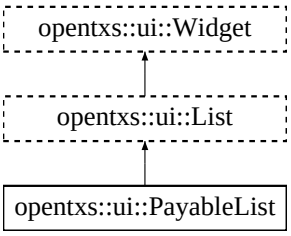
The documentation for this class was generated from the following file:

- include/opentxs/api/session/OTX.hpp

6.54 opentxs::ui::PayableList Class Reference

`#include <PayableList.hpp>`

Inheritance diagram for opentxs::ui::PayableList:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::PayableListItem >=0
  returns the first row, containing a valid PayableListItem or an empty smart pointer (if list is empty).
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::PayableListItem >=0
  returns the next row, containing a valid PayableListItem or an empty smart pointer (if at end of list).

6.54.1 Detailed Description

Like ContactList, this model manages a set of rows containing the Contacts in the wallet. However, this model only contains contacts that are payable. Each row contains a PayableListItem model, which is derived from ContactListItem.

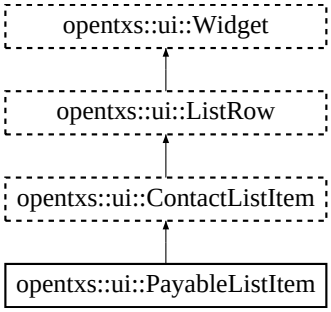
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/PayableList.hpp

6.55 opentxs::ui::PayableListItem Class Reference

`#include <PayableListItem.hpp>`

Inheritance diagram for opentxs::ui::PayableListItem:



Public Member Functions

- virtual auto **PaymentCode** () const noexcept -> UnallocatedCString=0

  Returns the payment code for this contact.

6.55.1 Detailed Description

This model represents a single PayableListItem from a row in the PayableList model. It contains everything found in a ContactListItem (from which it is derived) but it adds a method for retrieving the payment code for this contact.

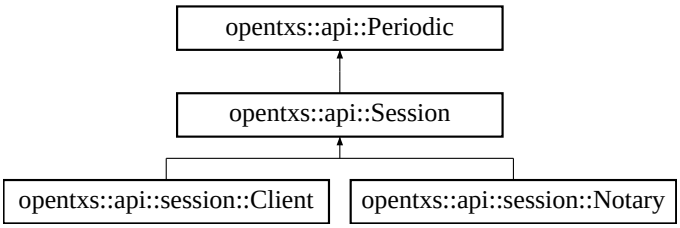
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/PayableListItem.hpp

6.56 opentxs::api::Periodic Class Reference

`#include <Periodic.hpp>`

Inheritance diagram for opentxs::api::Periodic:



Public Member Functions

- virtual auto **Cancel** (const int task) const -> bool=0

  Cancels a periodic task.
- virtual auto **Reschedule** (const int task, const std::chrono::seconds &interval) const -> bool=0

  Reschedules a periodic task.
- virtual auto Schedule (const std::chrono::seconds &interval, const opentxs::PeriodicTask &task, const std::chrono::seconds &last=0s) const -> int=0

6.56.1 Detailed Description

The Periodic API is used for scheduling and canceling recurring tasks.

6.56.2 Member Function Documentation

6.56.2.1 Schedule()

```
virtual auto opentxs::api::Periodic::Schedule (
            const std::chrono::seconds & interval,
            const opentxs::PeriodicTask & task,
            const std::chrono::seconds & last = 0s ) const -> int  [pure virtual]
```

Adds a task to the periodic task list with the specified interval. By default, schedules for immediate execution.

Returns

: task identifier which may be used to manage the task

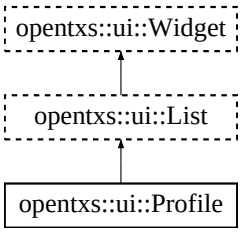
The documentation for this class was generated from the following file:

- include/opentxs/api/Periodic.hpp

6.57 opentxs::ui::Profile Class Reference

`#include <Profile.hpp>`

Inheritance diagram for opentxs::ui::Profile:



Public Types

- using **ItemType** = std::pair< identity::wot::claim::ClaimType, UnallocatedCString >
- using **ItemTypeList** = UnallocatedVector< ItemType >
- using **SectionType** = std::pair< identity::wot::claim::SectionType, UnallocatedCString >
- using **SectionTypeList** = UnallocatedVector< SectionType >

Public Member Functions

- virtual auto **AddClaim** (const identity::wot::claim::SectionType section, const identity::wot::claim::ClaimType type, const UnallocatedCString &value, const bool primary, const bool active) const noexcept -> bool=0

  Adds a new claim to the user's credentials. Returns success or failure.
- virtual auto **AllowedItems** (const identity::wot::claim::SectionType section, const UnallocatedCString &lang) const noexcept -> ItemTypeList=0

  Returns a list of allowed item types for a given section and language.
- virtual auto **AllowedSections** (const UnallocatedCString &lang) const noexcept -> SectionTypeList=0

  Returns a list of allowed sections for a given language.

- virtual auto **Delete** (const int section, const int type, const UnallocatedCString &claimID) const noexcept -> bool=0

  Deletes a claim from the user's credentials. Returns success or failure.
- virtual auto **DisplayName** () const noexcept -> UnallocatedCString=0

  Returns the display name for the user's profile in this wallet.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ProfileSection >=0

  Returns the first ProfileSection for this profile, containing metadata about the wallet user.
- virtual auto **ID** () const noexcept -> UnallocatedCString=0

  Returns the profile ID for the wallet user.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ProfileSection >=0

  Returns the next ProfileSection for this profile, containing metadata about the wallet user.
- virtual auto **PaymentCode** () const noexcept -> UnallocatedCString=0

  Returns the payment code for the wallet user.
- virtual auto **SetActive** (const int section, const int type, const UnallocatedCString &claimID, const bool active) const noexcept -> bool=0

  Sets a given claim as 'active' or 'inactive' in the user's credentials.
- virtual auto **SetPrimary** (const int section, const int type, const UnallocatedCString &claimID, const bool primary) const noexcept -> bool=0

  Sets a given claim as 'primary' or 'not primary' in the user's credentials. (Such as primary display name).
- virtual auto **SetValue** (const int section, const int type, const UnallocatedCString &claimID, const UnallocatedCString &value) const noexcept -> bool=0

  Sets the value for a given claim in this user's credentials.

6.57.1 Detailed Description

This model contains the wallet user's Profile data, and also has methods for manipulating that data. This includes the claims on this user's identity credentials.

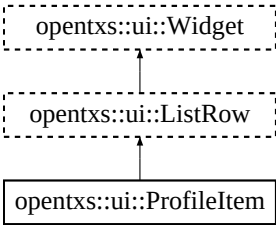
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/Profile.hpp

6.58 opentxs::ui::ProfileItem Class Reference

`#include <ProfileItem.hpp>`

Inheritance diagram for opentxs::ui::ProfileItem:



Public Member Functions

- virtual auto **ClaimID** () const noexcept -> UnallocatedCString=0

  Returns the ID for this claim.
- virtual auto **Delete** () const noexcept -> bool=0

  Deletes this claim from the user's credentials. Returns success or failure.
- virtual auto **IsActive** () const noexcept -> bool=0

  Indicates whether or not this claim is active.
- virtual auto **IsPrimary** () const noexcept -> bool=0

  Indicates whether or not this claim is primary.
- virtual auto **SetActive** (const bool &active) const noexcept -> bool=0

  Sets this claim as 'active' or 'inactive' on the wallet user's credentials.
- virtual auto **SetPrimary** (const bool &primary) const noexcept -> bool=0

  Sets this claim as 'primary' or 'not primary' on the wallet user's credentials.
- virtual auto **SetValue** (const UnallocatedCString &value) const noexcept -> bool=0
- virtual auto **Value** () const noexcept -> UnallocatedCString=0

  Returns the value of this claim.

6.58.1 Detailed Description

This model represents a single claim on the wallet user's identity credentials. Each of these claims is a single row on the ProfileSubsection model.

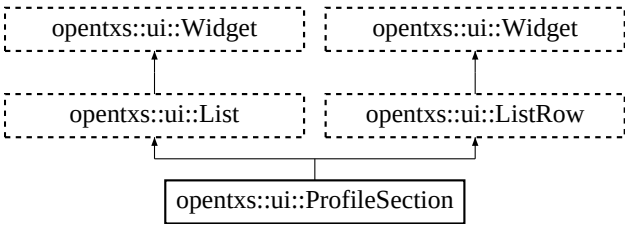
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ProfileItem.hpp

6.59 opentxs::ui::ProfileSection Class Reference

`#include <ProfileSection.hpp>`

Inheritance diagram for opentxs::ui::ProfileSection:



Public Types

- using **ItemType** = std::pair< identity::wot::claim::ClaimType, UnallocatedCString >
- using **ItemTypeList** = UnallocatedVector< ItemType >

Public Member Functions

- virtual auto **AddClaim** (const identity::wot::claim::ClaimType type, const UnallocatedCString &value, const bool primary, const bool active) const noexcept -> bool=0

  Used for adding a claim to this section of the user's credentials.
- virtual auto **Delete** (const int type, const UnallocatedCString &claimID) const noexcept -> bool=0

  Used for deleting a claim from this section of the user's credentials.
- virtual auto **Items** (const UnallocatedCString &lang) const noexcept -> ItemTypeList=0

  Returns a list of item types for a given language from this section of the user's credentials.
- virtual auto **Name** (const UnallocatedCString &lang) const noexcept -> UnallocatedCString=0

  Returns the display name for this section of the user's credentials.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ProfileSubsection >=0

  Returns the first ProfileSubsection for this section of the user's credentials.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ProfileSubsection >=0

  Returns the next ProfileSubsection for this section of the user's credentials.
- virtual auto **SetActive** (const int type, const UnallocatedCString &claimID, const bool active) const noexcept -> bool=0

  Sets a given claim as 'active' or 'inactive' in this section of the user's credentials.
- virtual auto **SetPrimary** (const int type, const UnallocatedCString &claimID, const bool primary) const noexcept -> bool=0

  Sets a given claim as 'primary' or 'not primary' in this section of the user's credentials.
- virtual auto **SetValue** (const int type, const UnallocatedCString &claimID, const UnallocatedCString &value) const noexcept -> bool=0

  Sets the contents for a given claim in this section of the user's credentials.
- virtual auto **Type** () const noexcept -> identity::wot::claim::SectionType=0

  Returns the SectionType for this ProfileSection.

Static Public Member Functions

- static auto **AllowedItems** (const identity::wot::claim::SectionType section, const UnallocatedCString &lang) noexcept -> ItemTypeList

  Returns a list of allowed item types for a given ProfileSection and language.

6.59.1 Detailed Description

This model represents a section of meta-data for the wallet user. Each row is a ProfileSubsection containing metadata about the wallet user from his identity credentials.

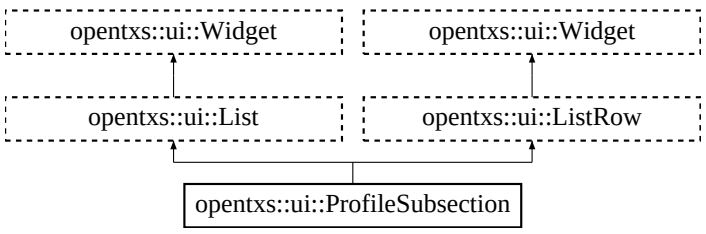
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ProfileSection.hpp

6.60 opentxs::ui::ProfileSubsection Class Reference

`#include <ProfileSubsection.hpp>`

Inheritance diagram for opentxs::ui::ProfileSubsection:



Public Member Functions

- virtual auto **AddItem** (const UnallocatedCString &value, const bool primary, const bool active) const noexcept -> bool=0

  Adds a new claim to this ProfileSubsection. Returns success or failure.
- virtual auto **Delete** (const UnallocatedCString &claimID) const noexcept -> bool=0

  Deletes a claim from this subsection of the user's credentials.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ProfileItem >=0

  Returns the first claim in this ProfileSubsection as a ProfileItem.
- virtual auto **Name** (const UnallocatedCString &lang) const noexcept -> UnallocatedCString=0

  Returns the display name of this ProfileSubsection for a given language.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::ProfileItem >=0

  Returns the next claim in this ProfileSubsection as a ProfileItem.
- virtual auto **SetActive** (const UnallocatedCString &claimID, const bool active) const noexcept -> bool=0

  Used to set a given claim in this subsection as 'active' or 'inactive'.
- virtual auto **SetPrimary** (const UnallocatedCString &claimID, const bool primary) const noexcept -> bool=0

  Used to set a given claim in this subsection as 'primary' or 'not primary'.
- virtual auto **SetValue** (const UnallocatedCString &claimID, const UnallocatedCString &value) const noexcept -> bool=0

  Sets the value for a given claimID in this subsection of the user's credentials.
- virtual auto **Type** () const noexcept -> identity::wot::claim::ClaimType=0

  Returns the ClaimType for this subsection of the user's credentials. All claims in this subsection are the same type.

6.60.1 Detailed Description

This model represents a subsection of meta-data for a ProfileSection for the wallet user Profile. Each row is a ProfileItem containing metadata about the wallet user from his identity credentials.

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/ProfileSubsection.hpp

6.61 opentxs::api::crypto::Seed Class Reference

`#include <Seed.hpp>`

Public Member Functions

- virtual auto **AccountChildKey** (const ReadView &path, const BIP44Chain internal, const Bip32Index index, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **AllowedSeedTypes** () const noexcept -> const UnallocatedMap< opentxs::crypto::SeedStyle, UnallocatedCString > &=0
- virtual auto **AllowedLanguages** (const opentxs::crypto::SeedStyle type) const noexcept -> const UnallocatedMap< opentxs::crypto::Language, UnallocatedCString > &=0
- virtual auto **AllowedSeedStrength** (const opentxs::crypto::SeedStyle type) const noexcept -> const UnallocatedMap< opentxs::crypto::SeedStrength, UnallocatedCString > &=0
- virtual auto **Bip32Root** (const UnallocatedCString &seedID, const PasswordPrompt &reason) const -> UnallocatedCString=0
- virtual auto **DefaultSeed** () const -> std::pair< UnallocatedCString, std::size_t >=0

- virtual auto **GetHDKey** (const UnallocatedCString &seedID, const opentxs::crypto::EcdsaCurve &curve, const UnallocatedVector< Bip32Index > &path, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **GetHDKey** (const UnallocatedCString &seedID, const opentxs::crypto::EcdsaCurve &curve, const UnallocatedVector< Bip32Index > &path, const opentxs::crypto::key::asymmetric::Role, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **GetHDKey** (const UnallocatedCString &seedID, const opentxs::crypto::EcdsaCurve &curve, const UnallocatedVector< Bip32Index > &path, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **GetHDKey** (const UnallocatedCString &seedID, const opentxs::crypto::EcdsaCurve &curve, const UnallocatedVector< Bip32Index > &path, const opentxs::crypto::key::asymmetric::Role, const VersionNumber version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::HD >=0
- virtual auto **GetPaymentCode** (const UnallocatedCString &seedID, const Bip32Index nym, const std::uint8_t version, const PasswordPrompt &reason) const -> std::unique_ptr< opentxs::crypto::key::Secp256k1 >=0
- virtual auto **GetStorageKey** (const UnallocatedCString &seedID, const PasswordPrompt &reason) const -> OTSymmetricKey=0
- virtual auto **GetSeed** (const UnallocatedCString &seedID, Bip32Index &index, const PasswordPrompt &reason) const -> OTSecret=0
- virtual auto **GetSeed** (const Identifier &id, const PasswordPrompt &reason) const noexcept -> opentxs::crypto::Seed=0
- virtual auto **ImportRaw** (const Secret &entropy, const PasswordPrompt &reason) const -> UnallocatedCString=0
- virtual auto **ImportSeed** (const Secret &words, const Secret &passphrase, const opentxs::crypto::SeedStyle type, const opentxs::crypto::Language lang, const PasswordPrompt &reason, const std::string_view comment={}) const -> UnallocatedCString=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Seed &=0
- virtual auto **LongestWord** (const opentxs::crypto::SeedStyle type, const opentxs::crypto::Language lang) const noexcept -> std::size_t=0
- virtual auto **NewSeed** (const opentxs::crypto::SeedStyle type, const opentxs::crypto::Language lang, const opentxs::crypto::SeedStrength strength, const PasswordPrompt &reason, const std::string_view comment={}) const -> UnallocatedCString=0
- virtual auto **Passphrase** (const UnallocatedCString &seedID, const PasswordPrompt &reason) const -> UnallocatedCString=0
- virtual auto **SeedDescription** (UnallocatedCString seedID) const noexcept -> UnallocatedCString=0
- virtual auto **SetDefault** (const Identifier &id) const noexcept -> bool=0
- virtual auto **SetSeedComment** (const Identifier &id, const std::string_view comment) const noexcept -> bool=0
- virtual auto **ValidateWord** (const opentxs::crypto::SeedStyle type, const opentxs::crypto::Language lang, const std::string_view word) const noexcept -> UnallocatedVector< UnallocatedCString >=0
- virtual auto **WordCount** (const opentxs::crypto::SeedStyle type, const opentxs::crypto::SeedStrength strength) const noexcept -> std::size_t=0
- virtual auto **Words** (const UnallocatedCString &seedID, const PasswordPrompt &reason) const -> UnallocatedCString=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Seed &=0

6.61.1 Detailed Description

The api::crypto::Seed API contains various seed-related crypto functions.

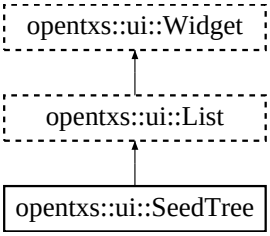
The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Seed.hpp

6.62 opentxs::ui::SeedTree Class Reference

`#include <SeedTree.hpp>`

Inheritance diagram for opentxs::ui::SeedTree:



Public Member Functions

- virtual auto **Debug** () const noexcept -> UnallocatedCString=0

  Returns debug information about the seeds in the wallet.
- virtual auto **DefaultNym** () const noexcept -> OTNymID=0

  Returns the ID for the wallet's default Nym.
- virtual auto **DefaultSeed** () const noexcept -> OTIdentifier=0

  Returns the ID for the wallet's default Seed.
- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< SeedTreeItem >=0

  Returns the first SeedTreeItem row.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< SeedTreeItem >=0

  Returns the next SeedTreeItem row.

6.62.1 Detailed Description

This model represents the list of seeds available in this wallet.

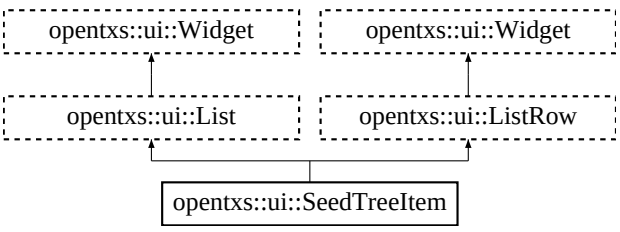
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/SeedTree.hpp

6.63 opentxs::ui::SeedTreeItem Class Reference

`#include <SeedTreeItem.hpp>`

Inheritance diagram for opentxs::ui::SeedTreeItem:



Public Member Functions

- virtual auto **Debug** () const noexcept -> UnallocatedCString=0

  Returns debug information about this item.
- virtual auto **First** () const noexcept -> SharedPimpl< SeedTreeNym >=0

  Returns the first nym for this seed, as a SeedTreeNym.
- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns the display name for this seed.
- virtual auto **Next** () const noexcept -> SharedPimpl< SeedTreeNym >=0

  Returns the next nym for this seed, as a SeedTreeNym.
- virtual auto **SeedID** () const noexcept -> UnallocatedCString=0

  Returns the ID for this seed.
- virtual auto **Type** () const noexcept -> crypto::SeedStyle=0

  Returns the seed type as an enum SeedStyle.

6.63.1 Detailed Description

This model represents one of the seeds available in this wallet. Each of the rows in the SeedTree model is a different SeedTreeItem.

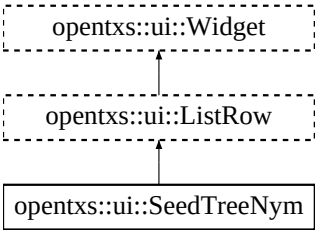
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/SeedTreeItem.hpp

6.64 opentxs::ui::SeedTreeNym Class Reference

`#include <SeedTreeNym.hpp>`

Inheritance diagram for opentxs::ui::SeedTreeNym:



Public Member Functions

- virtual auto **Index** () const noexcept -> std::size_t=0

  Returns the index for this Nym.
- virtual auto **Name** () const noexcept -> UnallocatedCString=0

  Returns the display name for this Nym.
- virtual auto **NymID** () const noexcept -> UnallocatedCString=0

  Returns the NymID for this Nym.

6.64.1 Detailed Description

This model represents one of the nyms available for a seed in this wallet. Each of the rows in the SeedTreeItem model is a different SeedTreeNym.

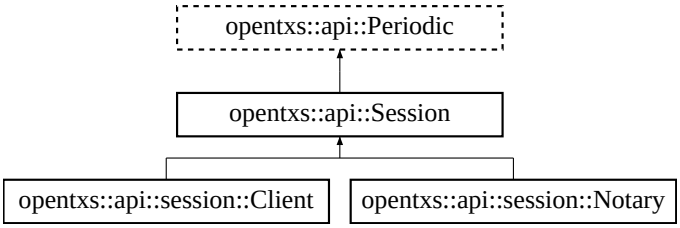
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/SeedTreeNym.hpp

6.65 opentxs::api::Session Class Reference

`#include <Session.hpp>`

Inheritance diagram for opentxs::api::Session:



Public Member Functions

- virtual auto **Config** () const noexcept -> const api::Settings &=0

  Returns a handle to the session-level config API.
- virtual auto **Crypto** () const noexcept -> const session::Crypto &=0

  Returns a handle to the session-level crypto API.
- virtual auto **DataFolder** () const noexcept -> const UnallocatedCString &=0

  Returns the data folder for this session.
- virtual auto **Endpoints** () const noexcept -> const session::Endpoints &=0

  Returns the Endpoints for this session.
- virtual auto **Factory** () const noexcept -> const session::Factory &=0

  Returns the Factory used for instantiating session objects.
- virtual auto **GetOptions** () const noexcept -> const Options &=0

  Returns an Options object.
- virtual auto **Instance** () const noexcept -> int=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const session::internal::Session &=0
- virtual auto **Network** () const noexcept -> const network::Network &=0

  Returns the network API for this session.
- virtual auto **QtRootObject** () const noexcept -> QObject *=0
- virtual auto **SetMasterKeyTimeout** (const std::chrono::seconds &timeout) const noexcept -> void=0

  This timeout determines how long the software will keep a master key available in memory.
- virtual OPENTXS_NO_EXPORT auto **Storage** () const noexcept -> const session::Storage &=0
- virtual auto **Wallet** () const noexcept -> const session::Wallet &=0

  Returns the Wallet API for this session.
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> session::internal::Session &=0

6.65.1 Detailed Description

This is the Session API, used for all client and server sessions.

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Session.hpp

6.66 opentxs::api::Settings Class Reference

`#include <Settings.hpp>`

Public Member Functions

- virtual void **SetConfigFilePath** (const String &strConfigFilePath) const =0
- virtual auto **HasConfigFilePath** () const -> bool=0
- virtual auto **IsLoaded** () const -> const Flag &=0

  Indicates whether or not the config file has been loaded.
- virtual auto **IsEmpty** () const -> bool=0

  Core (Reset Config, and Check if Config is empty)
- virtual auto CheckSetSection (const String &strSection, const String &strComment, bool &out_bIsNew↩ Section) const -> bool=0
- virtual auto **SetOption_bool** (const String &strSection, const String &strKey, bool &bVariableName) const -> bool=0

  Set Option helper function for setting bools.
- virtual auto **Reset** () -> bool=0

Load and Save.

Core (Public Load and Save)

Returns

Success or failure.

- virtual auto **Load** () const -> bool=0
- virtual auto **Save** () const -> bool=0

Check

Check Only (get value of key from configuration, if the key exists, then out_bKeyExist will be true.)

Returns

Success or failure.

- virtual auto **Check_str** (const String &strSection, const String &strKey, String &out_strResult, bool &out↩ _bKeyExist) const -> bool=0
- virtual auto **Check_long** (const String &strSection, const String &strKey, std::int64_t &out_lResult, bool &out_bKeyExist) const -> bool=0
- virtual auto **Check_bool** (const String &strSection, const String &strKey, bool &out_bResult, bool &out↩ _bKeyExist) const -> bool=0

Set only

Set new or update value, out_bNewOrUpdate will be true if the value changes.

Returns

Success or failure.

- virtual auto **Set_str** (const String &strSection, const String &strKey, const String &strValue, bool &out_↩ bNewOrUpdate) const -> bool=0
- virtual auto **Set_str** (const String &strSection, const String &strKey, const String &strValue, bool &out_↩ bNewOrUpdate, const String &strComment) const -> bool=0
- virtual auto **Set_long** (const String &strSection, const String &strKey, const std::int64_t &lValue, bool &out_bNewOrUpdate) const -> bool=0
- virtual auto **Set_long** (const String &strSection, const String &strKey, const std::int64_t &lValue, bool &out_bNewOrUpdate, const String &strComment) const -> bool=0
- virtual auto **Set_bool** (const String &strSection, const String &strKey, const bool &bValue, bool &out_b↩ NewOrUpdate) const -> bool=0
- virtual auto **Set_bool** (const String &strSection, const String &strKey, const bool &bValue, bool &out_b↩ NewOrUpdate, const String &strComment) const -> bool=0

- virtual auto CheckSet_str (const String &strSection, const String &strKey, const String &strDefault, UnallocatedCString &out_strResult, bool &out_bIsNew) const -> bool=0
- virtual auto **CheckSet_str** (const String &strSection, const String &strKey, const String &strDefault, UnallocatedCString &out_strResult, bool &out_bIsNew, const String &strComment) const -> bool=0
- virtual auto **CheckSet_str** (const String &strSection, const String &strKey, const String &strDefault, String &out_strResult, bool &out_bIsNew) const -> bool=0
- virtual auto **CheckSet_str** (const String &strSection, const String &strKey, const String &strDefault, String &out_strResult, bool &out_bIsNew, const String &strComment) const -> bool=0
- virtual auto **CheckSet_long** (const String &strSection, const String &strKey, const std::int64_t &lDefault, std::int64_t &out_lResult, bool &out_bIsNew) const -> bool=0
- virtual auto **CheckSet_long** (const String &strSection, const String &strKey, const std::int64_t &lDefault, std::int64_t &out_lResult, bool &out_bIsNew, const String &strComment) const -> bool=0
- virtual auto **CheckSet_bool** (const String &strSection, const String &strKey, const bool &bDefault, bool &out_bResult, bool &out_bIsNew) const -> bool=0
- virtual auto **CheckSet_bool** (const String &strSection, const String &strKey, const bool &bDefault, bool &out_bResult, bool &out_bIsNew, const String &strComment) const -> bool=0

6.66.1 Detailed Description

The Settings API, used for working with the config files.

6.66.2 Member Function Documentation

6.66.2.1 CheckSet_str()

```
virtual auto opentxs::api::Settings::CheckSet_str (
            const String & strSection,
            const String & strKey,
            const String & strDefault,
            UnallocatedCString & out_strResult,
            bool & out_bIsNew ) const -> bool  [pure virtual]
```

Check for Key, and returns if the key exists, otherwise will set the default key. If the default key is set, then out_b↩ IsNew will be true.)

6.66.2.2 CheckSetSection()

```
virtual auto opentxs::api::Settings::CheckSetSection (
            const String & strSection,
            const String & strComment,
            bool & out_bIsNewSection ) const -> bool  [pure virtual]
```

Check for a Section, if the section dosn't exist, it will be made and out_bIsNewSection will be true.)

The documentation for this class was generated from the following file:

- include/opentxs/api/Settings.hpp

6.67 opentxs::api::session::Storage Class Reference

Public Types

- using **Bip47ChannelList** = UnallocatedSet< OTIdentifier >

Public Member Functions

- virtual auto **AccountAlias** (const Identifier &accountID) const -> UnallocatedCString=0
- virtual auto **AccountList** () const -> ObjectList=0
- virtual auto **AccountContract** (const Identifier &accountID) const -> OTUnitID=0
- virtual auto **AccountIssuer** (const Identifier &accountID) const -> OTNymID=0
- virtual auto **AccountOwner** (const Identifier &accountID) const -> OTNymID=0
- virtual auto **AccountServer** (const Identifier &accountID) const -> OTNotaryID=0
- virtual auto **AccountSigner** (const Identifier &accountID) const -> OTNymID=0
- virtual auto **AccountUnit** (const Identifier &accountID) const -> UnitType=0
- virtual auto **AccountsByContract** (const identifier::UnitDefinition &contract) const -> UnallocatedSet< OTIdentifier >=0
- virtual auto **AccountsByIssuer** (const identifier::Nym &issuerNym) const -> UnallocatedSet< OTIdentifier >=0
- virtual auto **AccountsByOwner** (const identifier::Nym &ownerNym) const -> UnallocatedSet< OTIdentifier >=0
- virtual auto **AccountsByServer** (const identifier::Notary &server) const -> UnallocatedSet< OTIdentifier >=0
- virtual auto **AccountsByUnit** (const UnitType unit) const -> UnallocatedSet< OTIdentifier >=0
- virtual auto **Bip47Chain** (const identifier::Nym &nymID, const Identifier &channelID) const -> UnitType=0
- virtual auto **Bip47ChannelsByChain** (const identifier::Nym &nymID, const UnitType chain) const -> Bip47↩ ChannelList=0
- virtual auto **BlockchainAccountList** (const UnallocatedCString &nymID, const UnitType type) const -> UnallocatedSet< UnallocatedCString >=0
- virtual auto **BlockchainSubaccountAccountType** (const identifier::Nym &owner, const Identifier &id) const -> UnitType=0
- virtual auto **BlockchainThreadMap** (const identifier::Nym &nym, const Data &txid) const noexcept -> UnallocatedVector< OTIdentifier >=0
- virtual auto **BlockchainTransactionList** (const identifier::Nym &nym) const noexcept -> Unallocated↩ Vector< OTData >=0
- virtual auto **CheckTokenSpent** (const identifier::Notary &notary, const identifier::UnitDefinition &unit, const std::uint64_t series, const UnallocatedCString &key) const -> bool=0
- virtual auto **ContactAlias** (const UnallocatedCString &id) const -> UnallocatedCString=0

- virtual auto **ContactList** () const -> ObjectList=0
- virtual auto **ContextList** (const UnallocatedCString &nymID) const -> ObjectList=0
- virtual auto **ContactOwnerNym** (const UnallocatedCString &nymID) const -> UnallocatedCString=0
- virtual void **ContactSaveIndices** () const =0
- virtual auto **ContactUpgradeLevel** () const -> VersionNumber=0
- virtual auto **CreateThread** (const UnallocatedCString &nymID, const UnallocatedCString &threadID, const UnallocatedSet< UnallocatedCString > &participants) const -> bool=0
- virtual auto **DeleteAccount** (const UnallocatedCString &id) const -> bool=0
- virtual auto **DefaultNym** () const -> OTNymID=0
- virtual auto **DefaultSeed** () const -> UnallocatedCString=0
- virtual auto **DeleteContact** (const UnallocatedCString &id) const -> bool=0
- virtual auto **DeletePaymentWorkflow** (const UnallocatedCString &nymID, const UnallocatedCString &workflowID) const -> bool=0
- virtual auto **HashType** () const -> std::uint32_t=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Storage &=0
- virtual auto **IssuerList** (const UnallocatedCString &nymID) const -> ObjectList=0
- virtual auto **Load** (const UnallocatedCString &accountID, UnallocatedCString &output, UnallocatedCString &alias, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nymID, const UnallocatedCString &accountID, proto::↩ HDAccount &output, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::Nym &nymID, const Identifier &channelID, proto::Bip47Channel &output, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &id, proto::Contact &contact, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &id, proto::Contact &contact, UnallocatedCString &alias, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nym, const UnallocatedCString &id, proto::Context &context, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &id, proto::Credential &cred, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::Nym &id, proto::Nym &nym, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::Nym &id, proto::Nym &nym, UnallocatedCString &alias, const bool checking=false) const -> bool=0
- virtual auto **LoadNym** (const identifier::Nym &id, AllocateOutput destination, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nymID, const UnallocatedCString &id, proto::Issuer &issuer, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nymID, const UnallocatedCString &workflowID, proto::↩ PaymentWorkflow &workflow, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nymID, const UnallocatedCString &id, const otx::client::↩ StorageBox box, UnallocatedCString &output, UnallocatedCString &alias, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nymID, const UnallocatedCString &id, const otx::client::↩ StorageBox box, proto::PeerReply &request, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &nymID, const UnallocatedCString &id, const otx::client::↩ StorageBox box, proto::PeerRequest &request, std::time_t &time, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::Nym &nym, const identifier::Notary &notary, const identifier::UnitDefinition &unit, proto::Purse &output, const bool checking) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &id, proto::Seed &seed, const bool checking=false) const -> bool=0
- virtual auto **Load** (const UnallocatedCString &id, proto::Seed &seed, UnallocatedCString &alias, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::Notary &id, proto::ServerContract &contract, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::Notary &id, proto::ServerContract &contract, UnallocatedCString &alias, const bool checking=false) const -> bool=0

- virtual auto **Load** (const UnallocatedCString &nymId, const UnallocatedCString &threadId, proto::Storage↩ Thread &thread) const -> bool=0
- virtual auto **Load** (proto::Ciphertext &output, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::UnitDefinition &id, proto::UnitDefinition &contract, const bool checking=false) const -> bool=0
- virtual auto **Load** (const identifier::UnitDefinition &id, proto::UnitDefinition &contract, UnallocatedCString &alias, const bool checking=false) const -> bool=0
- virtual auto **LocalNyms** () const -> const UnallocatedSet< UnallocatedCString >=0
- virtual void **MapPublicNyms** (NymLambda &lambda) const =0
- virtual void **MapServers** (ServerLambda &lambda) const =0
- virtual void **MapUnitDefinitions** (UnitLambda &lambda) const =0
- virtual auto **MarkTokenSpent** (const identifier::Notary &notary, const identifier::UnitDefinition &unit, const std::uint64_t series, const UnallocatedCString &key) const -> bool=0
- virtual auto **MoveThreadItem** (const UnallocatedCString &nymId, const UnallocatedCString &fromThreadID, const UnallocatedCString &toThreadID, const UnallocatedCString &itemID) const -> bool=0
- virtual auto **NymBoxList** (const UnallocatedCString &nymID, const otx::client::StorageBox box) const -> ObjectList=0
- virtual auto **NymList** () const -> ObjectList=0
- virtual auto **PaymentWorkflowList** (const UnallocatedCString &nymID) const -> ObjectList=0
- virtual auto **PaymentWorkflowLookup** (const UnallocatedCString &nymID, const UnallocatedCString &sourceID) const -> UnallocatedCString=0
- virtual auto **PaymentWorkflowsByAccount** (const UnallocatedCString &nymID, const UnallocatedCString &accountID) const -> UnallocatedSet< UnallocatedCString >=0
- virtual auto **PaymentWorkflowsByState** (const UnallocatedCString &nymID, const otx::client::Payment↩ WorkflowType type, const otx::client::PaymentWorkflowState state) const -> UnallocatedSet< Unallocated↩ CString >=0
- virtual auto **PaymentWorkflowsByUnit** (const UnallocatedCString &nymID, const UnallocatedCString &unitID) const -> UnallocatedSet< UnallocatedCString >=0
- virtual auto **PaymentWorkflowState** (const UnallocatedCString &nymID, const UnallocatedCString &workflowID) const -> std::pair< otx::client::PaymentWorkflowType, otx::client::PaymentWorkflowState >=0
- virtual auto **RelabelThread** (const UnallocatedCString &threadID, const UnallocatedCString &label) const -> bool=0
- virtual auto **RemoveBlockchainThreadItem** (const identifier::Nym &nym, const Identifier &thread, const opentxs::blockchain::Type chain, const Data &txid) const noexcept -> bool=0
- virtual auto **RemoveNymBoxItem** (const UnallocatedCString &nymID, const otx::client::StorageBox box, const UnallocatedCString &itemID) const -> bool=0
- virtual auto **RemoveServer** (const UnallocatedCString &id) const -> bool=0
- virtual auto **RemoveThreadItem** (const identifier::Nym &nym, const Identifier &thread, const Unallocated↩ CString &id) const -> bool=0
- virtual auto **RemoveUnitDefinition** (const UnallocatedCString &id) const -> bool=0
- virtual auto **RenameThread** (const UnallocatedCString &nymId, const UnallocatedCString &threadId, const UnallocatedCString &newID) const -> bool=0
- virtual void **RunGC** () const =0
- virtual auto **ServerAlias** (const UnallocatedCString &id) const -> UnallocatedCString=0
- virtual auto **ServerList** () const -> ObjectList=0
- virtual auto **SeedList** () const -> ObjectList=0
- virtual auto **SetAccountAlias** (const UnallocatedCString &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto **SetContactAlias** (const UnallocatedCString &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto **SetDefaultNym** (const identifier::Nym &id) const -> bool=0
- virtual auto **SetDefaultSeed** (const UnallocatedCString &id) const -> bool=0
- virtual auto **SetNymAlias** (const identifier::Nym &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto **SetPeerRequestTime** (const UnallocatedCString &nymID, const UnallocatedCString &id, const otx::client::StorageBox box) const -> bool=0

- virtual auto **SetReadState** (const UnallocatedCString &nymId, const UnallocatedCString &threadId, const UnallocatedCString &itemId, const bool unread) const -> bool=0
- virtual auto **SetSeedAlias** (const UnallocatedCString &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto **SetServerAlias** (const identifier::Notary &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto **SetThreadAlias** (const UnallocatedCString &nymId, const UnallocatedCString &threadId, const UnallocatedCString &alias) const -> bool=0
- virtual auto **SetUnitDefinitionAlias** (const identifier::UnitDefinition &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto **Store** (const UnallocatedCString &accountID, const UnallocatedCString &data, const UnallocatedCString &alias, const identifier::Nym &ownerNym, const identifier::Nym &signerNym, const identifier::Nym &issuerNym, const identifier::Notary &server, const identifier::UnitDefinition &contract, const UnitType unit) const -> bool=0
- virtual auto **Store** (const UnallocatedCString &nymID, const opentxs::identity::wot::claim::ClaimType type, const proto::HDAccount &data) const -> bool=0
- virtual auto **Store** (const identifier::Nym &nymID, const Identifier &channelID, const proto::Bip47Channel &data) const -> bool=0
- virtual auto **Store** (const proto::Contact &data) const -> bool=0
- virtual auto **Store** (const proto::Context &data) const -> bool=0
- virtual auto **Store** (const proto::Credential &data) const -> bool=0
- virtual auto **Store** (const proto::Nym &data, const UnallocatedCString &alias={}) const -> bool=0
- virtual auto **Store** (const ReadView &data, const UnallocatedCString &alias={}) const -> bool=0
- virtual auto **Store** (const UnallocatedCString &nymID, const proto::Issuer &data) const -> bool=0
- virtual auto **Store** (const UnallocatedCString &nymID, const proto::PaymentWorkflow &data) const -> bool=0
- virtual auto **Store** (const UnallocatedCString &nymid, const UnallocatedCString &threadid, const UnallocatedCString &itemid, const std::uint64_t time, const UnallocatedCString &alias, const Unallocated↩ CString &data, const otx::client::StorageBox box, const UnallocatedCString &account={}) const -> bool=0
- virtual auto **Store** (const identifier::Nym &nym, const Identifier &thread, const opentxs::blockchain::Type chain, const Data &txid, const Time time) const noexcept -> bool=0
- virtual auto **Store** (const proto::PeerReply &data, const UnallocatedCString &nymid, const otx::client::↩ StorageBox box) const -> bool=0
- virtual auto **Store** (const proto::PeerRequest &data, const UnallocatedCString &nymid, const otx::client::↩ StorageBox box) const -> bool=0
- virtual auto **Store** (const identifier::Nym &nym, const proto::Purse &purse) const -> bool=0
- virtual auto **Store** (const proto::Seed &data) const -> bool=0
- virtual auto **Store** (const proto::ServerContract &data, const UnallocatedCString &alias={}) const -> bool=0
- virtual auto **Store** (const proto::Ciphertext &serialized) const -> bool=0
- virtual auto **Store** (const proto::UnitDefinition &data, const UnallocatedCString &alias={}) const -> bool=0
- virtual auto **ThreadList** (const UnallocatedCString &nymID, const bool unreadOnly) const -> ObjectList=0
- virtual auto **ThreadAlias** (const UnallocatedCString &nymID, const UnallocatedCString &threadID) const -> UnallocatedCString=0
- virtual auto **UnaffiliatedBlockchainTransaction** (const identifier::Nym &recipient, const Data &txid) const noexcept -> bool=0
- virtual auto **UnitDefinitionAlias** (const UnallocatedCString &id) const -> UnallocatedCString=0
- virtual auto **UnitDefinitionList** () const -> ObjectList=0
- virtual auto **UnreadCount** (const UnallocatedCString &nymId, const UnallocatedCString &threadId) const -> std::size_t=0
- virtual void **UpgradeNyms** ()=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Storage &=0

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Storage.hpp

6.68 opentxs::api::crypto::Symmetric Class Reference

`#include <Symmetric.hpp>`

Public Member Functions

- virtual OPENTXS_NO_EXPORT auto **InternalSymmetric** () const noexcept -> const internal::Symmetric &=0
- virtual auto **IvSize** (const opentxs::crypto::key::symmetric::Algorithm mode) const -> std::size_t=0
- virtual auto **Key** (const PasswordPrompt &password, const opentxs::crypto::key::symmetric::Algorithm mode=opentxs::crypto::key::symmetric::Algorithm::ChaCha20Poly1305) const -> OTSymmetricKey=0
- virtual auto **Key** (const ReadView &serializedCiphertext, const opentxs::crypto::key::symmetric::Algorithm mode) const -> OTSymmetricKey=0
- virtual auto **Key** (const Secret &seed, const std::uint64_t operations=0, const std::uint64_t difficulty=0, const opentxs::crypto::key::symmetric::Algorithm mode=opentxs::crypto::key::symmetric::Algorithm::ChaCha20↩ Poly1305, const opentxs::crypto::key::symmetric::Source type=opentxs::crypto::key::symmetric::Source::↩ Argon2i) const -> OTSymmetricKey=0
- virtual auto **Key** (const Secret &seed, const ReadView salt, const std::uint64_t operations, const std::uint64↩ _t difficulty, const std::uint64_t parallel, const std::size_t bytes, const opentxs::crypto::key::symmetric::Source type) const -> OTSymmetricKey=0
- virtual OPENTXS_NO_EXPORT auto **InternalSymmetric** () noexcept -> internal::Symmetric &=0

6.68.1 Detailed Description

The api::crypto::Symmetric API contains functions specific to symmetric keys. (AES etc).

The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Symmetric.hpp

6.69 opentxs::api::session::UI Class Reference

Public Member Functions

- virtual auto **AccountActivity** (const identifier::Nym &nymID, const Identifier &accountID, const Simple↩ Callback updateCB={}) const noexcept -> const opentxs::ui::AccountActivity &=0
- virtual auto **AccountActivityQt** (const identifier::Nym &nymID, const Identifier &accountID, const Simple↩ Callback updateCB={}) const noexcept -> opentxs::ui::AccountActivityQt *=0

  Caller does not own this pointer.
- virtual auto **AccountList** (const identifier::Nym &nym, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::AccountList &=0
- virtual auto **AccountListQt** (const identifier::Nym &nym, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::AccountListQt *=0

  Caller does not own this pointer.
- virtual auto **AccountSummary** (const identifier::Nym &nymID, const UnitType currency, const Simple↩ Callback updateCB={}) const noexcept -> const opentxs::ui::AccountSummary &=0
- virtual auto **AccountSummaryQt** (const identifier::Nym &nymID, const UnitType currency, const Simple↩ Callback updateCB={}) const noexcept -> opentxs::ui::AccountSummaryQt *=0

  Caller does not own this pointer.

- virtual auto **AccountTree** (const identifier::Nym &nym, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::AccountTree &=0
- virtual auto **AccountTreeQt** (const identifier::Nym &nym, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::AccountTreeQt *=0

  Caller does not own this pointer.
- virtual auto **ActivitySummary** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::ActivitySummary &=0
- virtual auto **ActivitySummaryQt** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::ActivitySummaryQt *=0

  Caller does not own this pointer.
- virtual auto **ActivityThread** (const identifier::Nym &nymID, const Identifier &threadID, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::ActivityThread &=0
- virtual auto **ActivityThreadQt** (const identifier::Nym &nymID, const Identifier &threadID, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::ActivityThreadQt *=0

  Caller does not own this pointer.
- virtual auto **BlankModel** (const std::size_t columns) const noexcept -> QAbstractItemModel *=0

  Caller does not own this pointer.
- virtual auto **BlockchainAccountStatus** (const identifier::Nym &nymID, const opentxs::blockchain::Type chain, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::BlockchainAccountStatus &=0
- virtual auto **BlockchainAccountStatusQt** (const identifier::Nym &nymID, const opentxs::blockchain::Type chain, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::BlockchainAccountStatusQt *=0

  Caller does not own this pointer.
- virtual auto **BlockchainIssuerID** (const opentxs::blockchain::Type chain) const noexcept -> const identifier::Nym &=0
- virtual auto **BlockchainNotaryID** (const opentxs::blockchain::Type chain) const noexcept -> const identifier::Notary &=0
- virtual auto **BlockchainSelection** (const opentxs::ui::Blockchains type, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::BlockchainSelection &=0
- virtual auto **BlockchainSelectionQt** (const opentxs::ui::Blockchains type, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::BlockchainSelectionQt *=0

  Caller does not own this pointer.
- virtual auto **BlockchainStatistics** (const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::BlockchainStatistics &=0
- virtual auto **BlockchainStatisticsQt** (const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::BlockchainStatisticsQt *=0
- virtual auto **BlockchainUnitID** (const opentxs::blockchain::Type chain) const noexcept -> const identifier::UnitDefinition &=0
- virtual auto **Contact** (const Identifier &contactID, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::Contact &=0
- virtual auto **ContactQt** (const Identifier &contactID, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::ContactQt *=0
- virtual auto **ContactList** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::ContactList &=0
- virtual auto **ContactListQt** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::ContactListQt *=0

  Caller does not own this pointer.
- virtual auto **IdentityManagerQt** () const noexcept -> opentxs::ui::IdentityManagerQt *=0

  Caller does not own this pointer.
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::UI &=0
- virtual auto **MessagableList** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::MessagableList &=0
- virtual auto **MessagableListQt** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::MessagableListQt *=0

  Caller does not own this pointer.

- virtual auto **NymList** (const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::NymList &=0
- virtual auto **NymListQt** (const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::NymListQt *=0

  Caller does not own this pointer.
- virtual auto **PayableList** (const identifier::Nym &nymID, const UnitType currency, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::PayableList &=0
- virtual auto **PayableListQt** (const identifier::Nym &nymID, const UnitType currency, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::PayableListQt *=0

  Caller does not own this pointer.
- virtual auto **Profile** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::Profile &=0
- virtual auto **ProfileQt** (const identifier::Nym &nymID, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::ProfileQt *=0

  Caller does not own this pointer.
- virtual auto **SeedTree** (const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::SeedTree &=0
- virtual auto **SeedTreeQt** (const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::SeedTreeQt *=0

  Caller does not own this pointer.
- virtual auto **SeedValidator** (const opentxs::crypto::SeedStyle type, const opentxs::crypto::Language lang) const noexcept -> const opentxs::ui::SeedValidator *=0

  Caller does not own this pointer.
- virtual auto **UnitList** (const identifier::Nym &nym, const SimpleCallback updateCB={}) const noexcept -> const opentxs::ui::UnitList &=0
- virtual auto **UnitListQt** (const identifier::Nym &nym, const SimpleCallback updateCB={}) const noexcept -> opentxs::ui::UnitListQt *=0

  Caller does not own this pointer.
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::UI &=0

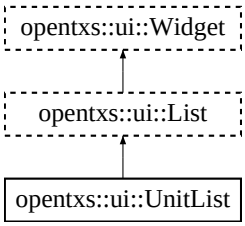
The documentation for this class was generated from the following file:

- include/opentxs/api/session/UI.hpp

6.70 opentxs::ui::UnitList Class Reference

`#include <UnitList.hpp>`

Inheritance diagram for opentxs::ui::UnitList:



Public Member Functions

- virtual auto **First** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::UnitListItem >=0

  Returns the first unit type available in this wallet as a UnitListItem.
- virtual auto **Next** () const noexcept -> opentxs::SharedPimpl< opentxs::ui::UnitListItem >=0

  Returns the next unit type available in this wallet as a UnitListItem.

6.70.1 Detailed Description

This model contains a row for each of the different unit types available in this wallet.

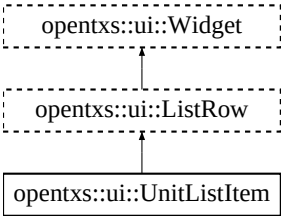
The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/UnitList.hpp

6.71 opentxs::ui::UnitListItem Class Reference

`#include <UnitListItem.hpp>`

Inheritance diagram for opentxs::ui::UnitListItem:



Public Member Functions

- virtual auto **Name** () const noexcept -> UnallocatedCString=0
  The display name of this unit type.
- virtual auto **Unit** () const noexcept -> UnitType=0
  The appropriate UnitType enum value for this unit type.

6.71.1 Detailed Description

This model describes a single unit type from the UnitList. (Bitcoin, Ethereum, etc).

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/UnitListItem.hpp

6.72 opentxs::api::crypto::Util Class Reference

`#include <Util.hpp>`

Public Member Functions

- virtual OPENTXS_NO_EXPORT auto **InternalUtil** () const noexcept -> const internal::Util &=0
- virtual auto **RandomizeMemory** (void *destination, const std::size_t size) const -> bool=0
- virtual OPENTXS_NO_EXPORT auto **InternalUtil** () noexcept -> internal::Util &=0

6.72.1 Detailed Description

the api::crypto::Util API contains utility functions specific to crypto.

The documentation for this class was generated from the following file:

- include/opentxs/api/crypto/Util.hpp

6.73 opentxs::api::session::Wallet Class Reference

This class manages instantiated contracts and provides easy access to them.

`#include <Wallet.hpp>`

Public Types

- using **AccountCallback** = std::function< void(const Account &)>

Public Member Functions

- virtual auto **AccountPartialMatch** (const UnallocatedCString &hint) const -> OTIdentifier=0
- virtual auto **DeleteAccount** (const Identifier &accountID) const -> bool=0
- virtual auto **UpdateAccount** (const Identifier &accountID, const otx::context::Server &, const String &serialized, const PasswordPrompt &reason) const -> bool=0
- virtual auto **UpdateAccount** (const Identifier &accountID, const otx::context::Server &, const String &serialized, const UnallocatedCString &label, const PasswordPrompt &reason) const -> bool=0
- virtual auto **ImportAccount** (std::unique_ptr< opentxs::Account > &imported) const -> bool=0
- virtual auto Context (const identifier::Notary &notaryID, const identifier::Nym &clientNymID) const -> std::shared_ptr< const otx::context::Base >=0
- virtual auto ClientContext (const identifier::Nym &remoteNymID) const -> std::shared_ptr< const otx::context::Client >=0
- virtual auto **DefaultNym** () const noexcept -> std::pair< OTNymID, std::size_t >=0
- virtual auto ServerContext (const identifier::Nym &localNymID, const Identifier &remoteID) const -> std::shared_ptr< const otx::context::Server >=0
- virtual auto IssuerList (const identifier::Nym &nymID) const -> UnallocatedSet< OTNymID >=0
- virtual auto **IsLocalNym** (const UnallocatedCString &id) const -> bool=0
- virtual auto **IsLocalNym** (const identifier::Nym &id) const -> bool=0
- virtual auto **LocalNymCount** () const -> std::size_t=0
- virtual auto **LocalNyms** () const -> UnallocatedSet< OTNymID >=0
- virtual auto Nym (const identifier::Nym &id, const std::chrono::milliseconds &timeout=0ms) const -> Nym_p=0
- virtual auto Nym (const ReadView &bytes) const -> Nym_p=0
- virtual auto **Nym** (const identity::Type type, const PasswordPrompt &reason, const UnallocatedCString &name={}) const -> Nym_p=0
- virtual auto **Nym** (const opentxs::crypto::Parameters &parameters, const PasswordPrompt &reason, const UnallocatedCString &name={}) const -> Nym_p=0
- virtual auto **Nym** (const PasswordPrompt &reason, const UnallocatedCString &name={}) const -> Nym_p=0
- virtual auto **Nym** (const opentxs::crypto::Parameters &parameters, const identity::Type type, const PasswordPrompt &reason, const UnallocatedCString &name={}) const -> Nym_p=0
- virtual auto **mutable_Nym** (const identifier::Nym &id, const PasswordPrompt &reason) const -> NymData=0

- virtual auto **Nymfile** (const identifier::Nym &id, const PasswordPrompt &reason) const -> std::unique_ptr< const opentxs::NymFile >=0
- virtual auto **NymByIDPartialMatch** (const UnallocatedCString &partialId) const -> Nym_p=0
- virtual auto NymList () const -> ObjectList=0
- virtual auto **NymNameByIndex** (const std::size_t index, String &name) const -> bool=0
- virtual auto PeerReply (const identifier::Nym &nym, const Identifier &reply, const otx::client::StorageBox &box, AllocateOutput destination) const -> bool=0
- virtual auto PeerReplyComplete (const identifier::Nym &nym, const Identifier &replyOrRequest) const -> bool=0
- virtual auto PeerReplyCreateRollback (const identifier::Nym &nym, const Identifier &request, const Identifier &reply) const -> bool=0
- virtual auto PeerReplySent (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerReplyIncoming (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerReplyFinished (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerReplyProcessed (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerReplyReceive (const identifier::Nym &nym, const PeerObject &reply) const -> bool=0
- virtual auto PeerRequest (const identifier::Nym &nym, const Identifier &request, const otx::client::StorageBox &box, std::time_t &time, AllocateOutput destination) const -> bool=0
- virtual auto PeerRequestComplete (const identifier::Nym &nym, const Identifier &reply) const -> bool=0
- virtual auto PeerRequestCreateRollback (const identifier::Nym &nym, const Identifier &request) const -> bool=0
- virtual auto PeerRequestDelete (const identifier::Nym &nym, const Identifier &request, const otx::client::↩ StorageBox &box) const -> bool=0
- virtual auto PeerRequestSent (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerRequestIncoming (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerRequestFinished (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerRequestProcessed (const identifier::Nym &nym) const -> ObjectList=0
- virtual auto PeerRequestReceive (const identifier::Nym &nym, const PeerObject &request) const -> bool=0
- virtual auto PeerRequestUpdate (const identifier::Nym &nym, const Identifier &request, const otx::client::↩ StorageBox &box) const -> bool=0
- virtual auto **Purse** (const identifier::Nym &nym, const identifier::Notary &server, const identifier::UnitDefinition &unit, const bool checking=false) const -> const otx::blind::Purse &=0
- virtual auto RemoveServer (const identifier::Notary &id) const -> bool=0
- virtual auto RemoveUnitDefinition (const identifier::UnitDefinition &id) const -> bool=0
- virtual auto Server (const identifier::Notary &id, const std::chrono::milliseconds &timeout=std::chrono↩ ::milliseconds(0)) const noexcept(false) -> OTServerContract=0
- virtual auto Server (const ReadView &contract) const noexcept(false) -> OTServerContract=0
- virtual auto Server (const UnallocatedCString &nymid, const UnallocatedCString &name, const UnallocatedCString &terms, const UnallocatedList< contract::Server::Endpoint > &endpoints, const PasswordPrompt &reason, const VersionNumber version) const noexcept(false) -> OTServerContract=0
- virtual auto ServerList () const -> ObjectList=0
- virtual auto **SetDefaultNym** (const identifier::Nym &id) const noexcept -> bool=0
- virtual auto SetNymAlias (const identifier::Nym &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto SetServerAlias (const identifier::Notary &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto SetUnitDefinitionAlias (const identifier::UnitDefinition &id, const UnallocatedCString &alias) const -> bool=0
- virtual auto UnitDefinitionList () const -> ObjectList=0
- virtual auto UnitDefinition (const identifier::UnitDefinition &id, const std::chrono::milliseconds &timeout=std↩ ::chrono::milliseconds(0)) const noexcept(false) -> OTUnitDefinition=0
- virtual auto UnitDefinition (const ReadView contract) const noexcept(false) -> OTUnitDefinition=0
- virtual auto **BasketContract** (const identifier::UnitDefinition &id, const std::chrono::milliseconds &timeout=std::chrono::milliseconds(0)) const noexcept(false) -> OTBasketContract=0
- virtual auto CurrencyContract (const UnallocatedCString &nymid, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const Amount &redemptionIncrement, const PasswordPrompt &reason) const noexcept(false) -> OTUnitDefinition=0

- virtual auto **CurrencyContract** (const UnallocatedCString &nymid, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const Amount &redemptionIncrement, const display::Definition &displayDefinition, const PasswordPrompt &reason) const noexcept(false) -> OTUnitDefinition=0
- virtual auto **CurrencyContract** (const UnallocatedCString &nymid, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const Amount &redemptionIncrement, const VersionNumber version, const PasswordPrompt &reason) const noexcept(false) -> OTUnitDefinition=0
- virtual auto **CurrencyContract** (const UnallocatedCString &nymid, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const Amount &redemptionIncrement, const display::Definition &displayDefinition, const VersionNumber version, const PasswordPrompt &reason) const noexcept(false) -> OTUnitDefinition=0
- virtual auto SecurityContract (const UnallocatedCString &nymid, const UnallocatedCString &shortname, const UnallocatedCString &terms, const UnitType unitOfAccount, const PasswordPrompt &reason, const display::Definition &displayDefinition, const Amount &redemptionIncrement, const VersionNumber version=contract::Unit::DefaultVersion) const noexcept(false) -> OTUnitDefinition=0
- virtual auto **CurrencyTypeBasedOnUnitType** (const identifier::UnitDefinition &contractID) const -> UnitType=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const session::internal::Wallet &=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> session::internal::Wallet &=0

6.73.1 Detailed Description

This class manages instantiated contracts and provides easy access to them.

It includes functionality which was previously found in OTWallet, and adds new capabilities such as the ability to (optionally) automatically perform remote lookups for contracts which are not already present in the local database.

6.73.2 Member Function Documentation

6.73.2.1 ClientContext()

```
virtual auto opentxs::api::session::Wallet::ClientContext (
            const identifier::Nym & remoteNymID ) const -> std::shared_ptr< const otx::context↩
::Client >  [pure virtual]
```

Load a read-only copy of a ClientContext object

Parameters

in	*remoteNymID*	context identifier (usually the other party's nym id)

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.73.2.2 Context()

```
virtual auto opentxs::api::session::Wallet::Context (
            const identifier::Notary & notaryID,
            const identifier::Nym & clientNymID ) const -> std::shared_ptr< const otx::context↩
::Base >  [pure virtual]
```

Load a read-only copy of a Context object

This method should only be called if the specific client or server version is not available (such as by classes common to client and server).

Parameters

in	*notaryID*	
in	*clientNymID*	

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.73.2.3 CurrencyContract()

```
virtual auto opentxs::api::session::Wallet::CurrencyContract (
            const UnallocatedCString & nymid,
            const UnallocatedCString & shortname,
            const UnallocatedCString & terms,
            const UnitType unitOfAccount,
            const Amount & redemptionIncrement,
            const PasswordPrompt & reason ) const -> OTUnitDefinition  [pure virtual], [noexcept]
```

Create a new currency contract

Parameters

in	*nymid*	the identifier of nym which will create the contract
in	*shortname*	a short human-readable identifier for the contract
in	*terms*	human-readable terms and conditions

Exceptions

std::runtime_error	the contract can not be created

6.73.2.4 IssuerList()

```
virtual auto opentxs::api::session::Wallet::IssuerList (
```

```
            const identifier::Nym & nymID ) const -> UnallocatedSet< OTNymID >  [pure virtual]
```

Returns a list of all issuers associated with a local nym

6.73.2.5 Nym() [1/2]

```
virtual auto opentxs::api::session::Wallet::Nym (
            const identifier::Nym & id,
            const std::chrono::milliseconds & timeout = 0ms ) const -> Nym_p  [pure virtual]
```

Obtain a smart pointer to an instantiated nym.

The smart pointer will not be initialized if the object does not exist or is invalid.

If the caller is willing to accept a network lookup delay, it can specify a timeout to be used in the event that the contract can not be located in local storage and must be queried from a remote location.

If no timeout is specified, the remote query will still happen in the background, but this method will return immediately with a null result.

Parameters

in	*id*	the identifier of the nym to be returned
in	*timeout*	The caller can set a non-zero value here if it's willing to wait for a network lookup. The default value of 0 will return immediately.

6.73.2.6 Nym() [2/2]

```
virtual auto opentxs::api::session::Wallet::Nym (
            const ReadView & bytes ) const -> Nym_p  [pure virtual]
```

Instantiate a nym from serialized form

The smart pointer will not be initialized if the provided serialized contract is invalid.

Parameters

in	*nym*	the serialized version of the contract

6.73.2.7 NymList()

```
virtual auto opentxs::api::session::Wallet::NymList ( ) const -> ObjectList  [pure virtual]
```

Returns a list of all known nyms and their aliases

6.73.2.8 PeerReply()

```
virtual auto opentxs::api::session::Wallet::PeerReply (
             const identifier::Nym & nym,
             const Identifier & reply,
             const otx::client::StorageBox & box,
             AllocateOutput destination ) const -> bool  [pure virtual]
```

Load a peer reply object

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the peer reply object
in	*box*	the box from which to retrive the peer object

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.73.2.9 PeerReplyComplete()

```
virtual auto opentxs::api::session::Wallet::PeerReplyComplete (
             const identifier::Nym & nym,
             const Identifier & replyOrRequest ) const -> bool  [pure virtual]
```

Clean up the recipient's copy of a peer reply

The peer reply is moved from the nym's SentPeerReply box to the FinishedPeerReply box.

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*replyOrRequest*	the identifier of the peer reply object, or the id of its corresponding request

Returns

true if the request is successfully stored

6.73.2.10 PeerReplyCreateRollback()

```
virtual auto opentxs::api::session::Wallet::PeerReplyCreateRollback (
             const identifier::Nym & nym,
             const Identifier & request,
             const Identifier & reply ) const -> bool  [pure virtual]
```

Rollback a PeerReplyCreate call

The original request is returned to IncomingPeerRequest box

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the corresponding request
in	*reply*	the identifier of the peer reply object

Returns

true if the rollback is successful

6.73.2.11 PeerReplyFinished()

```
virtual auto opentxs::api::session::Wallet::PeerReplyFinished (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of finished peer replies

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.12 PeerReplyIncoming()

```
virtual auto opentxs::api::session::Wallet::PeerReplyIncoming (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of incoming peer replies

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.13 PeerReplyProcessed()

```
virtual auto opentxs::api::session::Wallet::PeerReplyProcessed (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of processed peer replies

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.14 PeerReplyReceive()

```
virtual auto opentxs::api::session::Wallet::PeerReplyReceive (
            const identifier::Nym & nym,
            const PeerObject & reply ) const -> bool  [pure virtual]
```

Store the senders's copy of a peer reply

The peer reply is stored in the IncomingPeerReply box for the specified nym.

The corresponding request is moved from the nym's SentPeerRequest box to the FinishedPeerRequest box.

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the corresponding request
in	*reply*	the serialized peer reply object

Returns

true if the request is successfully stored

6.73.2.15 PeerReplySent()

```
virtual auto opentxs::api::session::Wallet::PeerReplySent (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of sent peer replies

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.16 PeerRequest()

```
virtual auto opentxs::api::session::Wallet::PeerRequest (
            const identifier::Nym & nym,
```

```
            const Identifier & request,
            const otx::client::StorageBox & box,
            std::time_t & time,
            AllocateOutput destination ) const -> bool  [pure virtual]
```

Load a peer reply object

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the peer reply object
in	*box*	the box from which to retrive the peer object

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.73.2.17 PeerRequestComplete()

```
virtual auto opentxs::api::session::Wallet::PeerRequestComplete (
            const identifier::Nym & nym,
            const Identifier & reply ) const -> bool  [pure virtual]
```

Clean up the sender's copy of a peer reply

The peer reply is moved from the nym's IncomingPeerReply box to the ProcessedPeerReply box.

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*reply*	the identifier of the peer reply object

Returns

true if the request is successfully moved

6.73.2.18 PeerRequestCreateRollback()

```
virtual auto opentxs::api::session::Wallet::PeerRequestCreateRollback (
            const identifier::Nym & nym,
            const Identifier & request ) const -> bool  [pure virtual]
```

Rollback a PeerRequestCreate call

The request is deleted from to SentPeerRequest box

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the peer request

Returns

true if the rollback is successful

6.73.2.19 PeerRequestDelete()

```
virtual auto opentxs::api::session::Wallet::PeerRequestDelete (
            const identifier::Nym & nym,
            const Identifier & request,
            const otx::client::StorageBox & box ) const -> bool  [pure virtual]
```

Delete a peer reply object

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the peer reply object
in	*box*	the box from which the peer object will be deleted

6.73.2.20 PeerRequestFinished()

```
virtual auto opentxs::api::session::Wallet::PeerRequestFinished (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of finished peer requests

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.21 PeerRequestIncoming()

```
virtual auto opentxs::api::session::Wallet::PeerRequestIncoming (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of incoming peer requests

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.22 PeerRequestProcessed()

```
virtual auto opentxs::api::session::Wallet::PeerRequestProcessed (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of processed peer requests

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.23 PeerRequestReceive()

```
virtual auto opentxs::api::session::Wallet::PeerRequestReceive (
            const identifier::Nym & nym,
            const PeerObject & request ) const -> bool  [pure virtual]
```

Store the recipient's copy of a peer request

The peer request is stored in the IncomingPeerRequest box for the specified nym.

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the serialized peer request object

Returns

true if the request is successfully stored

6.73.2.24 PeerRequestSent()

```
virtual auto opentxs::api::session::Wallet::PeerRequestSent (
            const identifier::Nym & nym ) const -> ObjectList  [pure virtual]
```

Obtain a list of sent peer requests

Parameters

in	*nym*	the identifier of the nym whose box is returned

6.73.2.25 PeerRequestUpdate()

```
virtual auto opentxs::api::session::Wallet::PeerRequestUpdate (
            const identifier::Nym & nym,
            const Identifier & request,
            const otx::client::StorageBox & box ) const -> bool  [pure virtual]
```

Update the timestamp of a peer request object

Parameters

in	*nym*	the identifier of the nym who owns the object
in	*request*	the identifier of the peer request object
in	*box*	the box from which the peer object will be deleted

6.73.2.26 RemoveServer()

```
virtual auto opentxs::api::session::Wallet::RemoveServer (
            const identifier::Notary & id ) const -> bool  [pure virtual]
```

Unload and delete a server contract

This method destroys the contract object, removes it from the in-memory map, and deletes it from local storage.

Parameters

in	*id*	the indentifier of the contract to be removed

Returns

true if successful, false if the contract did not exist

6.73.2.27 RemoveUnitDefinition()

```
virtual auto opentxs::api::session::Wallet::RemoveUnitDefinition (
            const identifier::UnitDefinition & id ) const -> bool  [pure virtual]
```

Unload and delete a unit definition contract

This method destroys the contract object, removes it from the in-memory map, and deletes it from local storage.

Parameters

in	*id*	the indentifier of the contract to be removed

Returns

true if successful, false if the contract did not exist

6.73.2.28 SecurityContract()

```
virtual auto opentxs::api::session::Wallet::SecurityContract (
            const UnallocatedCString & nymid,
            const UnallocatedCString & shortname,
            const UnallocatedCString & terms,
            const UnitType unitOfAccount,
            const PasswordPrompt & reason,
            const display::Definition & displayDefinition,
            const Amount & redemptionIncrement,
            const VersionNumber version = contract::Unit::DefaultVersion ) const -> OTUnit↩
Definition  [pure virtual], [noexcept]
```

Create a new security contract

Parameters

in	*nymid*	the identifier of nym which will create the contract
in	*shortname*	a short human-readable identifier for the contract
in	*terms*	human-readable terms and conditions

Exceptions

std::runtime_error	the contract can not be created

6.73.2.29 Server() [1/3]

```
virtual auto opentxs::api::session::Wallet::Server (
            const identifier::Notary & id,
            const std::chrono::milliseconds & timeout = std::chrono::milliseconds(0) ) const
-> OTServerContract  [pure virtual], [noexcept]
```

Obtain an instantiated server contract.

If the caller is willing to accept a network lookup delay, it can specify a timeout to be used in the event that the contract can not be located in local storage and must be queried from a remote location.

If no timeout is specified, the remote query will still happen in the background, but this method will return immediately with a null result.

Parameters

in	*id*	the identifier of the contract to be returned
in	*timeout*	The caller can set a non-zero value here if it's willing to wait for a network lookup. The default value of 0 will return immediately.

Exceptions

std::runtime_error	the specified contract does not exist in the wallet

6.73.2.30 Server() [2/3]

```
virtual auto opentxs::api::session::Wallet::Server (
            const ReadView & contract ) const -> OTServerContract  [pure virtual], [noexcept]
```

Instantiate a server contract from serialized form

Parameters

in	*contract*	the serialized version of the contract

Exceptions

std::runtime_error	the provided contract is not valid

6.73.2.31 Server() [3/3]

```
virtual auto opentxs::api::session::Wallet::Server (
            const UnallocatedCString & nymid,
            const UnallocatedCString & name,
            const UnallocatedCString & terms,
            const UnallocatedList< contract::Server::Endpoint > & endpoints,
            const PasswordPrompt & reason,
            const VersionNumber version ) const -> OTServerContract  [pure virtual], [noexcept]
```

Create a new server contract

Parameters

in	*nymid*	the identifier of nym which will create the contract
in	*name*	the official name of the server
in	*terms*	human-readable server description & terms of use
in	*url*	externally-reachable IP address or hostname
in	*port*	externally-reachable listen port

Exceptions

std::runtime_error	the contract can not be created

6.73.2.32 ServerContext()

```
virtual auto opentxs::api::session::Wallet::ServerContext (
            const identifier::Nym & localNymID,
            const Identifier & remoteID ) const -> std::shared_ptr< const otx::context::↩
Server >  [pure virtual]
```

Load a read-only copy of a ServerContext object

Parameters

in	*localNymID*	the identifier of the nym who owns the context
in	*remoteID*	context identifier (usually the other party's nym id)

Returns

A smart pointer to the object. The smart pointer will not be instantiated if the object does not exist or is invalid.

6.73.2.33 ServerList()

```
virtual auto opentxs::api::session::Wallet::ServerList ( ) const -> ObjectList  [pure virtual]
```

Returns a list of all available server contracts and their aliases

6.73.2.34 SetNymAlias()

```
virtual auto opentxs::api::session::Wallet::SetNymAlias (
            const identifier::Nym & id,
            const UnallocatedCString & alias ) const -> bool  [pure virtual]
```

Updates the alias for the specified nym.

An alias is a local label which is not part of the nym credentials itself.

Parameters

in	*id*	the identifier of the nym whose alias is to be set
in	*alias*	the alias to set or update for the specified nym

Returns

true if successful, false if the nym can not be located

6.73.2.35 SetServerAlias()

```
virtual auto opentxs::api::session::Wallet::SetServerAlias (
            const identifier::Notary & id,
            const UnallocatedCString & alias ) const -> bool  [pure virtual]
```

Updates the alias for the specified server contract.

An alias is a local label which is not part of the server contract itself.

Parameters

in	*id*	the identifier of the contract whose alias is to be set
in	*alias*	the alias to set or update for the specified contract

Returns

true if successful, false if the contract can not be located

6.73.2.36 SetUnitDefinitionAlias()

```
virtual auto opentxs::api::session::Wallet::SetUnitDefinitionAlias (
            const identifier::UnitDefinition & id,
            const UnallocatedCString & alias ) const -> bool  [pure virtual]
```

Updates the alias for the specified unit definition contract.

An alias is a local label which is not part of the unit definition contract itself.

Parameters

in	*id*	the identifier of the contract whose alias is to be set
in	*alias*	the alias to set or update for the specified contract

Returns

true if successful, false if the contract can not be located

6.73.2.37 UnitDefinition() [1/2]

```
virtual auto opentxs::api::session::Wallet::UnitDefinition (
            const identifier::UnitDefinition & id,
            const std::chrono::milliseconds & timeout = std::chrono::milliseconds(0) ) const
-> OTUnitDefinition  [pure virtual], [noexcept]
```

Obtain an instantiated unit definition contract.

If the caller is willing to accept a network lookup delay, it can specify a timeout to be used in the event that the contract can not be located in local storage and must be queried from a remote location.

If no timeout is specified, the remote query will still happen in the background, but this method will return immediately with a null result.

Parameters

in	*id*	the identifier of the contract to be returned
in	*timeout*	The caller can set a non-zero value here if it's willing to wait for a network lookup. The default value of 0 will return immediately.

Exceptions

std::runtime_error	the specified contract does not exist in the wallet

6.73.2.38 UnitDefinition() [2/2]

```
virtual auto opentxs::api::session::Wallet::UnitDefinition (
            const ReadView contract ) const -> OTUnitDefinition  [pure virtual], [noexcept]
```

Instantiate a unit definition contract from serialized form

Parameters

in	*contract*	the protobuf serialized version of the contract

Exceptions

std::runtime_error	the provided contract is invalid

6.73.2.39 UnitDefinitionList()

```
virtual auto opentxs::api::session::Wallet::UnitDefinitionList ( ) const -> ObjectList  [pure
virtual]
```

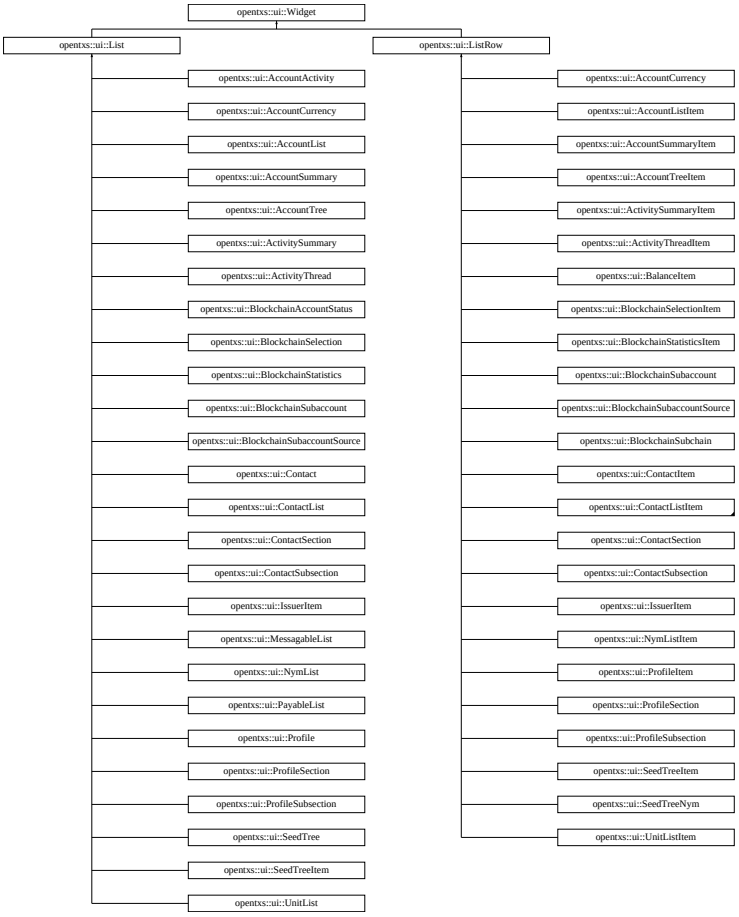
Obtain a list of all available unit definition contracts and their aliases

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Wallet.hpp

6.74 opentxs::ui::Widget Class Reference

Inheritance diagram for opentxs::ui::Widget:



Public Member Functions

- virtual void **ClearCallbacks** () const noexcept=0
- virtual void **SetCallback** (SimpleCallback cb) const noexcept=0
- virtual auto **WidgetID** () const noexcept -> OTIdentifier=0

The documentation for this class was generated from the following file:

- include/opentxs/interface/ui/Widget.hpp

6.75 opentxs::api::session::Workflow Class Reference

`#include <Workflow.hpp>`

Public Types

- using **Cheque** = std::pair< otx::client::PaymentWorkflowState, std::unique_ptr< opentxs::Cheque > >
- using **Purse** = std::pair< otx::client::PaymentWorkflowState, otx::blind::Purse >
- using **Transfer** = std::pair< otx::client::PaymentWorkflowState, std::unique_ptr< opentxs::Item > >

Public Member Functions

- virtual auto AbortTransfer (const identifier::Nym &nymID, const Item &transfer, const Message &reply) const -> bool=0
- virtual auto AcceptTransfer (const identifier::Nym &nymID, const identifier::Notary &notaryID, const OTTransaction &pending, const Message &reply) const -> bool=0
- virtual auto AcknowledgeTransfer (const identifier::Nym &nymID, const Item &transfer, const Message &reply) const -> bool=0
- virtual auto **AllocateCash** (const identifier::Nym &id, const otx::blind::Purse &purse) const -> OTIdentifier=0
- virtual auto CancelCheque (const opentxs::Cheque &cheque, const Message &request, const Message *reply) const -> bool=0
- virtual auto ClearCheque (const identifier::Nym &recipientNymID, const OTTransaction &receipt) const -> bool=0
- virtual auto ClearTransfer (const identifier::Nym &nymID, const identifier::Notary &notaryID, const OTTransaction &receipt) const -> bool=0
- virtual auto CompleteTransfer (const identifier::Nym &nymID, const identifier::Notary &notaryID, const OTTransaction &receipt, const Message &reply) const -> bool=0
- virtual auto ConveyTransfer (const identifier::Nym &nymID, const identifier::Notary &notaryID, const OTTransaction &pending) const -> OTIdentifier=0
- virtual auto CreateTransfer (const Item &transfer, const Message &request) const -> OTIdentifier=0
- virtual auto DepositCheque (const identifier::Nym &nymID, const Identifier &accountID, const opentxs::Cheque &cheque, const Message &request, const Message *reply) const -> bool=0
- virtual auto ExpireCheque (const identifier::Nym &nymID, const opentxs::Cheque &cheque) const -> bool=0
- virtual auto ExportCheque (const opentxs::Cheque &cheque) const -> bool=0
- virtual auto FinishCheque (const opentxs::Cheque &cheque, const Message &request, const Message *reply) const -> bool=0
- virtual auto ImportCheque (const identifier::Nym &nymID, const opentxs::Cheque &cheque) const -> OTIdentifier=0
- virtual auto **InstantiateCheque** (const identifier::Nym &nymID, const Identifier &workflowID) const -> Cheque=0
- virtual auto **InstantiatePurse** (const identifier::Nym &nymID, const Identifier &workflowID) const -> Purse=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () const noexcept -> const internal::Workflow &=0
- virtual auto **List** (const identifier::Nym &nymID, const otx::client::PaymentWorkflowType type, const otx::client::PaymentWorkflowState state) const -> UnallocatedSet< OTIdentifier >=0
- virtual auto **LoadCheque** (const identifier::Nym &nymID, const Identifier &chequeID) const -> Cheque=0
- virtual auto **LoadChequeByWorkflow** (const identifier::Nym &nymID, const Identifier &workflowID) const -> Cheque=0
- virtual auto **LoadTransfer** (const identifier::Nym &nymID, const Identifier &transferID) const -> Transfer=0
- virtual auto **LoadTransferByWorkflow** (const identifier::Nym &nymID, const Identifier &workflowID) const -> Transfer=0
- virtual OPENTXS_NO_EXPORT auto LoadWorkflow (const identifier::Nym &nymID, const Identifier &workflowID, proto::PaymentWorkflow &out) const -> bool=0

- virtual auto **ReceiveCash** (const identifier::Nym &receiver, const otx::blind::Purse &purse, const Message &message) const -> OTIdentifier=0
- virtual auto ReceiveCheque (const identifier::Nym &nymID, const opentxs::Cheque &cheque, const Message &message) const -> OTIdentifier=0
- virtual auto **SendCash** (const identifier::Nym &sender, const identifier::Nym &recipient, const Identifier &workflowID, const Message &request, const Message *reply) const -> bool=0
- virtual auto SendCheque (const opentxs::Cheque &cheque, const Message &request, const Message *reply) const -> bool=0
- virtual auto **WorkflowParty** (const identifier::Nym &nymID, const Identifier &workflowID, const int index) const -> const UnallocatedCString=0
- virtual auto **WorkflowPartySize** (const identifier::Nym &nymID, const Identifier &workflowID, int &partysize) const -> bool=0
- virtual auto **WorkflowState** (const identifier::Nym &nymID, const Identifier &workflowID) const -> otx↩ ::client::PaymentWorkflowState=0
- virtual auto **WorkflowType** (const identifier::Nym &nymID, const Identifier &workflowID) const -> otx::client↩ ::PaymentWorkflowType=0
- virtual auto WorkflowsByAccount (const identifier::Nym &nymID, const Identifier &accountID) const -> UnallocatedVector< OTIdentifier >=0
- virtual auto WriteCheque (const opentxs::Cheque &cheque) const -> OTIdentifier=0
- virtual OPENTXS_NO_EXPORT auto **Internal** () noexcept -> internal::Workflow &=0

Static Public Member Functions

- static OPENTXS_NO_EXPORT auto **ContainsCash** (const proto::PaymentWorkflow &workflow) -> bool
- static OPENTXS_NO_EXPORT auto **ContainsCheque** (const proto::PaymentWorkflow &workflow) -> bool
- static OPENTXS_NO_EXPORT auto **ContainsTransfer** (const proto::PaymentWorkflow &workflow) -> bool
- static OPENTXS_NO_EXPORT auto **ExtractCheque** (const proto::PaymentWorkflow &workflow) -> UnallocatedCString
- static OPENTXS_NO_EXPORT auto **ExtractPurse** (const proto::PaymentWorkflow &workflow, proto::Purse &out) -> bool
- static OPENTXS_NO_EXPORT auto **ExtractTransfer** (const proto::PaymentWorkflow &workflow) -> UnallocatedCString
- static OPENTXS_NO_EXPORT auto **InstantiateCheque** (const api::Session &api, const proto::Payment↩ Workflow &workflow) -> Cheque
- static OPENTXS_NO_EXPORT auto **InstantiatePurse** (const api::Session &api, const proto::Payment↩ Workflow &workflow) -> Purse
- static OPENTXS_NO_EXPORT auto **InstantiateTransfer** (const api::Session &api, const proto::Payment↩ Workflow &workflow) -> Transfer
- static OPENTXS_NO_EXPORT auto **UUID** (const api::Session &api, const proto::PaymentWorkflow &workflown) -> OTIdentifier
- static auto **UUID** (const api::Session &api, const Identifier &notary, const TransactionNumber &number) -> OTIdentifier

6.75.1 Detailed Description

Store and retrieve payment workflow events

Sequence for sending a cheque:

1. WriteCheque
2. SendCheque or ExportCheque a. SendCheque may be repeated as necessary until successful

3. (optional) ExpireCheque a. May be performed after step 1 or after step 2 b. It's possible that a cheque that's been expired locally was already accepted by the notary
4. (optional) CancelCheque a. May be performed after step 1 or after step 2
5. ClearCheque a. Called after receiving a cheque deposit receipt
6. FinishCheque a. Called after a process inbox attempt b. May be repeated as necessary until successful

Sequence for receiving a cheque:

1. ReceiveCheque or ImportCheque
2. (optional) ExpireCheque
3. DepositCheque a. May be repeated as necessary until successful

Sequence for sending a transfer:

1. CreateTransfer a. Called just before sending a notarizeTransaction message to notary.
2. AcknowledgeTransfer a. Called after receiving a server response with a successful transaction reply
3. AbortTransfer a. Called after receiving a server response with an unsuccessful transaction reply, or after proving the server did not receive the original transaction
4. ClearTransfer a. Called after receiving a transfer receipt
5. CompleteTransfer a. Called after a process inbox attempt b. May be repeated as necessary until successful

Sequence for receiving a transfer:

1. ConveyTransfer a. Called after receiving a transfer
2. AcceptTransfer a. May be repeated as necessary until successful

Sequence for an internal transfer: NOTE: AcknowledgeTransfer and ConveyTransfer may be called out of order

1. CreateTransfer a. Called just before sending a notarizeTransaction message to notary.
2. AcknowledgeTransfer a. Called after receiving a server response with a successful transaction reply
3. AbortTransfer a. Called after receiving a server response with an unsuccessful transaction reply, or after proving the server did not receive the original transaction
4. ConveyTransfer a. Called after receiving a transfer
5. ClearTransfer a. Called after receiving a transfer receipt
6. CompleteTransfer a. Called after a process inbox attempt b. May be repeated as necessary until successful

Sequence for sending cash

1. AllocateCash
2. SendCash

Sequence for receiving cash

1. ReceiveCash
2. AcceptCash

1. RejectCash

6.75.2 Member Function Documentation

6.75.2.1 AbortTransfer()

```
virtual auto opentxs::api::session::Workflow::AbortTransfer (
            const identifier::Nym & nymID,
            const Item & transfer,
            const Message & reply ) const -> bool  [pure virtual]
```

Record a failed transfer attempt

6.75.2.2 AcceptTransfer()

```
virtual auto opentxs::api::session::Workflow::AcceptTransfer (
            const identifier::Nym & nymID,
            const identifier::Notary & notaryID,
            const OTTransaction & pending,
            const Message & reply ) const -> bool  [pure virtual]
```

Record a transfer accept, or accept attempt

6.75.2.3 AcknowledgeTransfer()

```
virtual auto opentxs::api::session::Workflow::AcknowledgeTransfer (
            const identifier::Nym & nymID,
            const Item & transfer,
            const Message & reply ) const -> bool  [pure virtual]
```

Record a successful transfer attempt

6.75.2.4 CancelCheque()

```
virtual auto opentxs::api::session::Workflow::CancelCheque (
            const opentxs::Cheque & cheque,
            const Message & request,
            const Message * reply ) const -> bool  [pure virtual]
```

Record a cheque cancellation or cancellation attempt

6.75.2.5 ClearCheque()

```
virtual auto opentxs::api::session::Workflow::ClearCheque (
            const identifier::Nym & recipientNymID,
            const OTTransaction & receipt ) const -> bool  [pure virtual]
```

Record a cheque deposit receipt

6.75.2.6 ClearTransfer()

```
virtual auto opentxs::api::session::Workflow::ClearTransfer (
            const identifier::Nym & nymID,
            const identifier::Notary & notaryID,
            const OTTransaction & receipt ) const -> bool  [pure virtual]
```

Record receipt of a transfer receipt

6.75.2.7 CompleteTransfer()

```
virtual auto opentxs::api::session::Workflow::CompleteTransfer (
            const identifier::Nym & nymID,
            const identifier::Notary & notaryID,
            const OTTransaction & receipt,
            const Message & reply ) const -> bool  [pure virtual]
```

Record a process inbox for sender that accepts a transfer receipt

6.75.2.8 ConveyTransfer()

```
virtual auto opentxs::api::session::Workflow::ConveyTransfer (
            const identifier::Nym & nymID,
            const identifier::Notary & notaryID,
            const OTTransaction & pending ) const -> OTIdentifier  [pure virtual]
```

Create a new incoming transfer workflow, or update an existing internal transfer workflow.

6.75.2.9 CreateTransfer()

```
virtual auto opentxs::api::session::Workflow::CreateTransfer (
            const Item & transfer,
            const Message & request ) const -> OTIdentifier  [pure virtual]
```

Record a new outgoing or internal "sent transfer" (or attempt) workflow

6.75.2.10 DepositCheque()

```
virtual auto opentxs::api::session::Workflow::DepositCheque (
            const identifier::Nym & nymID,
            const Identifier & accountID,
            const opentxs::Cheque & cheque,
            const Message & request,
            const Message * reply ) const -> bool  [pure virtual]
```

Record a cheque deposit or deposit attempt

6.75.2.11 ExpireCheque()

```
virtual auto opentxs::api::session::Workflow::ExpireCheque (
            const identifier::Nym & nymID,
            const opentxs::Cheque & cheque ) const -> bool  [pure virtual]
```

Mark a cheque workflow as expired

6.75.2.12 ExportCheque()

```
virtual auto opentxs::api::session::Workflow::ExportCheque (
            const opentxs::Cheque & cheque ) const -> bool  [pure virtual]
```

Record an out of band cheque conveyance

6.75.2.13 FinishCheque()

```
virtual auto opentxs::api::session::Workflow::FinishCheque (
            const opentxs::Cheque & cheque,
            const Message & request,
            const Message * reply ) const -> bool  [pure virtual]
```

Record a process inbox that accepts a cheque deposit receipt

6.75.2.14 ImportCheque()

```
virtual auto opentxs::api::session::Workflow::ImportCheque (
            const identifier::Nym & nymID,
            const opentxs::Cheque & cheque ) const -> OTIdentifier  [pure virtual]
```

Create a new incoming cheque workflow from an out of band cheque

6.75.2.15 LoadWorkflow()

```
virtual OPENTXS_NO_EXPORT auto opentxs::api::session::Workflow::LoadWorkflow (
            const identifier::Nym & nymID,
            const Identifier & workflowID,
            proto::PaymentWorkflow & out ) const -> bool  [pure virtual]
```

Load a serialized workflow, if it exists

6.75.2.16 ReceiveCheque()

```
virtual auto opentxs::api::session::Workflow::ReceiveCheque (
            const identifier::Nym & nymID,
            const opentxs::Cheque & cheque,
            const Message & message ) const -> OTIdentifier  [pure virtual]
```

Create a new incoming cheque workflow from an OT message

6.75.2.17 SendCheque()

```
virtual auto opentxs::api::session::Workflow::SendCheque (
            const opentxs::Cheque & cheque,
            const Message & request,
            const Message * reply ) const -> bool  [pure virtual]
```

Record a send or send attempt via an OT notary

6.75.2.18 WorkflowsByAccount()

```
virtual auto opentxs::api::session::Workflow::WorkflowsByAccount (
            const identifier::Nym & nymID,
            const Identifier & accountID ) const -> UnallocatedVector< OTIdentifier >  [pure
virtual]
```

Get a list of workflow IDs relevant to a specified account

6.75.2.19 WriteCheque()

```
virtual auto opentxs::api::session::Workflow::WriteCheque (
            const opentxs::Cheque & cheque ) const -> OTIdentifier  [pure virtual]
```

Create a new outgoing cheque workflow

The documentation for this class was generated from the following file:

- include/opentxs/api/session/Workflow.hpp

6.76 opentxs::api::network::ZAP Class Reference

`#include <ZAP.hpp>`

Public Types

- using **Callback** = opentxs::network::zeromq::zap::Callback::ReceiveCallback
- using **Policy** = opentxs::network::zeromq::zap::Callback::Policy

Public Member Functions

- virtual auto RegisterDomain (const UnallocatedCString &domain, const Callback &callback) const -> bool=0
- virtual auto SetDefaultPolicy (const Policy policy) const -> bool=0

6.76.1 Detailed Description

api::network::ZAP used for accessing ZAP specific functionality.

6.76.2 Member Function Documentation

6.76.2.1 RegisterDomain()

```
virtual auto opentxs::api::network::ZAP::RegisterDomain (
            const UnallocatedCString & domain,
            const Callback & callback ) const -> bool  [pure virtual]
```

Set a callback that will be triggered for any ZAP requests in a specified domain

Parameters

in	*domain*	The ZAP domain to be registered. Any non-empty string is valid.
in	*callback*	The callback to be executed for the specified domain.

Returns

True if the domain is valid and not already registered

6.76.2.2 SetDefaultPolicy()

```
virtual auto opentxs::api::network::ZAP::SetDefaultPolicy (
            const Policy policy ) const -> bool  [pure virtual]
```

Configure ZAP policy for unhandled domains

Default behavior is Accept.

Parameters

in	*policy*	Accept or reject ZAP requests for a domain which has no registered callback

The documentation for this class was generated from the following file:

- include/opentxs/api/network/ZAP.hpp

6.77 opentxs::api::network::ZMQ Class Reference

```
#include <ZMQ.hpp>
```

Public Member Functions

- virtual auto **Context** () const -> const opentxs::network::zeromq::Context &=0
- virtual auto **DefaultAddressType** () const -> AddressType=0
- virtual auto **KeepAlive** () const -> std::chrono::seconds=0
- virtual void **KeepAlive** (const std::chrono::seconds duration) const =0
- virtual auto **Linger** () const -> std::chrono::seconds=0
- virtual auto **ReceiveTimeout** () const -> std::chrono::seconds=0
- virtual auto **Running** () const -> const Flag &=0
- virtual void **RefreshConfig** () const =0
- virtual auto **SendTimeout** () const -> std::chrono::seconds=0
- virtual auto **Server** (const UnallocatedCString &id) const -> opentxs::network::ServerConnection &=0
- virtual auto **SetSocksProxy** (const UnallocatedCString &proxy) const -> bool=0
- virtual auto **SocksProxy** () const -> UnallocatedCString=0
- virtual auto **SocksProxy** (UnallocatedCString &proxy) const -> bool=0
- virtual auto **Status** (const UnallocatedCString &server) const -> opentxs::network::ConnectionState=0

6.77.1 Detailed Description

api::network::ZMQ API used for accessing ZMQ-specific network functionality.

The documentation for this class was generated from the following file:

- include/opentxs/api/network/ZMQ.hpp

Chapter 7

File Documentation

7.1 Context.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <functional>
12
13 #include "opentxs/api/Periodic.hpp"
14 #include "opentxs/util/Bytes.hpp"
15 #include "opentxs/util/Container.hpp"
16
17 class QObject;
18
19 // NOLINTBEGIN(modernize-concat-nested-namespaces)
20 namespace opentxs  // NOLINT
21 {
22 // inline namespace v1
23 // {
24 namespace api
25 {
26 namespace internal
27 {
28 class Context;
29 }  // namespace internal
30
31 namespace network
32 {
33 class Asio;
34 class ZAP;
35 }  // namespace network
36
37 namespace session
38 {
39 class Client;
40 class Notary;
41 }  // namespace session
42
43 class Context;
44 class Crypto;
45 class Factory;
46 class Settings;
47 }  // namespace api
48
49 namespace network
50 {
51 namespace zeromq
52 {
53 class Context;
54 }  // namespace zeromq
55 }  // namespace network
56
57 namespace rpc
58 {
```

```
59 namespace request
60 {
61 class Base;
62 }  // namespace request
63
64 namespace response
65 {
66 class Base;
67 }  // namespace response
68 }  // namespace rpc
69
70 class Options;
71 // }  // namespace v1
72 }  // namespace opentxs
73 // NOLINTEND(modernize-concat-nested-namespaces)
74
79 class OPENTXS_EXPORT opentxs::api::Context : virtual public Periodic
80 {
81 public:
82     using ShutdownCallback = std::function<void()>;
83
86     static auto PrepareSignalHandling() noexcept -> void;
87     static auto SuggestFolder(const UnallocatedCString& app) noexcept
88         -> UnallocatedCString;
89
91     virtual auto Asio() const noexcept -> const network::Asio& = 0;
93     virtual auto ClientSession(const int instance) const noexcept(false)
94         -> const api::session::Client& = 0;
96     virtual auto ClientSessionCount() const noexcept -> std::size_t = 0;
98     virtual auto Config(const UnallocatedCString& path) const noexcept
99         -> const api::Settings& = 0;
101     virtual auto Crypto() const noexcept -> const api::Crypto& = 0;
103     virtual auto Factory() const noexcept -> const api::Factory& = 0;
106     virtual auto HandleSignals(
107         ShutdownCallback* callback = nullptr) const noexcept -> void = 0;
108     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
109         -> const internal::Context& = 0;
111     virtual auto NotarySession(const int instance) const noexcept(false)
112         -> const session::Notary& = 0;
114     virtual auto NotarySessionCount() const noexcept -> std::size_t = 0;
115     virtual auto ProfileId() const noexcept -> UnallocatedCString = 0;
116     OPENTXS_NO_EXPORT virtual auto QtRootObject() const noexcept
117         -> QObject* = 0;
119     virtual auto RPC(const rpc::request::Base& command) const noexcept
120         -> std::unique_ptr<rpc::response::Base> = 0;
121     virtual auto RPC(const ReadView command, const AllocateOutput response)
122         const noexcept -> bool = 0;
129     virtual auto StartClientSession(const Options& args, const int instance)
130         const -> const api::session::Client& = 0;
131     virtual auto StartClientSession(const int instance) const
132         -> const api::session::Client& = 0;
133     virtual auto StartClientSession(
134         const Options& args,
135         const int instance,
136         const UnallocatedCString& recoverWords,
137         const UnallocatedCString& recoverPassphrase) const
138         -> const api::session::Client& = 0;
145     virtual auto StartNotarySession(const Options& args, const int instance)
146         const -> const session::Notary& = 0;
147     virtual auto StartNotarySession(const int instance) const
148         -> const session::Notary& = 0;
150     virtual auto ZAP() const noexcept -> const api::network::ZAP& = 0;
152     virtual auto ZMQ() const noexcept
153         -> const opentxs::network::zeromq::Context& = 0;
154
155     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
156         -> internal::Context& = 0;
157
158     OPENTXS_NO_EXPORT ~Context() override = default;
159
160 protected:
161     Context() = default;
162
163 private:
164     Context(const Context&) = delete;
165     Context(Context&&) = delete;
166     auto operator=(const Context&) -> Context& = delete;
167     auto operator=(Context&&) -> Context& = delete;
168 };
```

7.2 Asymmetric.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
```

```
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <memory>
11
12 #include "opentxs/crypto/Bip32.hpp"
13 #include "opentxs/crypto/Types.hpp"
14 #include "opentxs/crypto/key/Types.hpp"
15 #include "opentxs/util/Numbers.hpp"
16
18 namespace opentxs  // NOLINT
19 {
20 // inline namespace v1
21 // {
22 namespace api
23 {
24 namespace crypto
25 {
26 namespace internal
27 {
28 class Asymmetric;
29 }  // namespace internal
30 }  // namespace crypto
31 }  // namespace api
32
33 namespace crypto
34 {
35 namespace key
36 {
37 class Asymmetric;
38 class HD;
39 class Secp256k1;
40 }  // namespace key
41
42 class Parameters;
43 }  // namespace crypto
44
45 class PasswordPrompt;
46 class Secret;
47 // }  // namespace v1
48 }  // namespace opentxs
49 // NOLINTEND(modernize-concat-nested-namespaces)
50
51 namespace opentxs::api::crypto
52 {
56 class OPENTXS_EXPORT Asymmetric
57 {
58 public:
59     virtual auto InstantiateKey(
60         const opentxs::crypto::key::asymmetric::Algorithm type,
61         const UnallocatedCString& seedID,
62         const opentxs::crypto::Bip32::Key& serialized,
63         const PasswordPrompt& reason) const noexcept
64         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
65     virtual auto InstantiateKey(
66         const opentxs::crypto::key::asymmetric::Algorithm type,
67         const UnallocatedCString& seedID,
68         const opentxs::crypto::Bip32::Key& serialized,
69         const opentxs::crypto::key::asymmetric::Role role,
70         const PasswordPrompt& reason) const noexcept
71         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
72     virtual auto InstantiateKey(
73         const opentxs::crypto::key::asymmetric::Algorithm type,
74         const UnallocatedCString& seedID,
75         const opentxs::crypto::Bip32::Key& serialized,
76         const VersionNumber version,
77         const PasswordPrompt& reason) const noexcept
78         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
79     virtual auto InstantiateKey(
80         const opentxs::crypto::key::asymmetric::Algorithm type,
81         const UnallocatedCString& seedID,
82         const opentxs::crypto::Bip32::Key& serialized,
83         const opentxs::crypto::key::asymmetric::Role role,
84         const VersionNumber version,
85         const PasswordPrompt& reason) const noexcept
86         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
87     virtual auto InstantiateSecp256k1Key(
88         const ReadView publicKey,
89         const PasswordPrompt& reason) const noexcept
90         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
91     virtual auto InstantiateSecp256k1Key(
92         const ReadView publicKey,
```

```
93          const opentxs::crypto::key::asymmetric::Role role,
94          const PasswordPrompt& reason) const noexcept
95          -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
96      virtual auto InstantiateSecp256k1Key(
97          const ReadView publicKey,
98          const VersionNumber version,
99          const PasswordPrompt& reason) const noexcept
100         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
101     virtual auto InstantiateSecp256k1Key(
102         const ReadView publicKey,
103         const opentxs::crypto::key::asymmetric::Role role,
104         const VersionNumber version,
105         const PasswordPrompt& reason) const noexcept
106         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
107     virtual auto InstantiateSecp256k1Key(
108         const Secret& privateKey,
109         const PasswordPrompt& reason) const noexcept
110         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
111     virtual auto InstantiateSecp256k1Key(
112         const Secret& privateKey,
113         const opentxs::crypto::key::asymmetric::Role role,
114         const PasswordPrompt& reason) const noexcept
115         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
116     virtual auto InstantiateSecp256k1Key(
117         const Secret& privateKey,
118         const VersionNumber version,
119         const PasswordPrompt& reason) const noexcept
120         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
121     virtual auto InstantiateSecp256k1Key(
122         const Secret& privateKey,
123         const opentxs::crypto::key::asymmetric::Role role,
124         const VersionNumber version,
125         const PasswordPrompt& reason) const noexcept
126         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
127     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
128         -> const internal::Asymmetric& = 0;
129     virtual auto NewHDKey(
130         const UnallocatedCString& seedID,
131         const Secret& seed,
132         const opentxs::crypto::EcdsaCurve& curve,
133         const opentxs::crypto::Bip32::Path& path,
134         const PasswordPrompt& reason) const
135         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
136     virtual auto NewHDKey(
137         const UnallocatedCString& seedID,
138         const Secret& seed,
139         const opentxs::crypto::EcdsaCurve& curve,
140         const opentxs::crypto::Bip32::Path& path,
141         const opentxs::crypto::key::asymmetric::Role role,
142         const PasswordPrompt& reason) const
143         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
144     virtual auto NewHDKey(
145         const UnallocatedCString& seedID,
146         const Secret& seed,
147         const opentxs::crypto::EcdsaCurve& curve,
148         const opentxs::crypto::Bip32::Path& path,
149         const VersionNumber version,
150         const PasswordPrompt& reason) const
151         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
152     virtual auto NewHDKey(
153         const UnallocatedCString& seedID,
154         const Secret& seed,
155         const opentxs::crypto::EcdsaCurve& curve,
156         const opentxs::crypto::Bip32::Path& path,
157         const opentxs::crypto::key::asymmetric::Role role,
158         const VersionNumber version,
159         const PasswordPrompt& reason) const
160         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
161     virtual auto NewKey(
162         const opentxs::crypto::Parameters& params,
163         const PasswordPrompt& reason) const
164         -> std::unique_ptr<opentxs::crypto::key::Asymmetric> = 0;
165     virtual auto NewKey(
166         const opentxs::crypto::Parameters& params,
167         const opentxs::crypto::key::asymmetric::Role role,
168         const PasswordPrompt& reason) const
169         -> std::unique_ptr<opentxs::crypto::key::Asymmetric> = 0;
170     virtual auto NewKey(
171         const opentxs::crypto::Parameters& params,
172         const VersionNumber version,
173         const PasswordPrompt& reason) const
174         -> std::unique_ptr<opentxs::crypto::key::Asymmetric> = 0;
175     virtual auto NewKey(
176         const opentxs::crypto::Parameters& params,
177         const opentxs::crypto::key::asymmetric::Role role,
178         const VersionNumber version,
179         const PasswordPrompt& reason) const
```

```
180         -> std::unique_ptr<opentxs::crypto::key::Asymmetric> = 0;
181     virtual auto NewSecp256k1Key(
182         const UnallocatedCString& seedID,
183         const Secret& seed,
184         const opentxs::crypto::Bip32::Path& path,
185         const PasswordPrompt& reason) const
186         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
187     virtual auto NewSecp256k1Key(
188         const UnallocatedCString& seedID,
189         const Secret& seed,
190         const opentxs::crypto::Bip32::Path& path,
191         const opentxs::crypto::key::asymmetric::Role role,
192         const PasswordPrompt& reason) const
193         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
194     virtual auto NewSecp256k1Key(
195         const UnallocatedCString& seedID,
196         const Secret& seed,
197         const opentxs::crypto::Bip32::Path& path,
198         const VersionNumber version,
199         const PasswordPrompt& reason) const
200         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
201     virtual auto NewSecp256k1Key(
202         const UnallocatedCString& seedID,
203         const Secret& seed,
204         const opentxs::crypto::Bip32::Path& path,
205         const opentxs::crypto::key::asymmetric::Role role,
206         const VersionNumber version,
207         const PasswordPrompt& reason) const
208         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
209
210     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
211         -> internal::Asymmetric& = 0;
212
213     OPENTXS_NO_EXPORT virtual ~Asymmetric() = default;
214
215 protected:
216     Asymmetric() = default;
217
218 private:
219     Asymmetric(const Asymmetric&) = delete;
220     Asymmetric(Asymmetric&&) = delete;
221     auto operator=(const Asymmetric&) -> Asymmetric& = delete;
222     auto operator=(Asymmetric&&) -> Asymmetric& = delete;
223 };
224 }  // namespace opentxs::api::crypto
```

7.3 Blockchain.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 // IWYU pragma: no_include "opentxs/blockchain/crypto/HDProtocol.hpp"
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include <cstdint>
13 #include <memory>
14
15 #include "opentxs/blockchain/Types.hpp"
16 #include "opentxs/blockchain/block/Types.hpp"
17 #include "opentxs/blockchain/crypto/Types.hpp"
18 #include "opentxs/core/Data.hpp"
19 #include "opentxs/core/identifier/Generic.hpp"
20 #include "opentxs/core/identifier/Nym.hpp"
21 #include "opentxs/crypto/Types.hpp"
22 #include "opentxs/util/Bytes.hpp"
23 #include "opentxs/util/Container.hpp"
24 #include "opentxs/util/Time.hpp"
25
26 // NOLINTBEGIN(modernize-concat-nested-namespaces)
27 namespace opentxs  // NOLINT
28 {
29 // inline namespace v1
30 // {
31 namespace api
32 {
33 namespace crypto
34 {
35 namespace internal
```

```
36 {
37 class Blockchain;
38 }  // namespace internal
39 }  // namespace crypto
40 }  // namespace api
41
42 namespace blockchain
43 {
44 namespace block
45 {
46 namespace bitcoin
47 {
48 class Transaction;
49 }  // namespace bitcoin
50 }  // namespace block
51
52 namespace crypto
53 {
54 class Account;
55 class Element;
56 class HD;
57 class PaymentCode;
58 class Wallet;
59 }  // namespace crypto
60
61 namespace node
62 {
63 class Manager;
64 }  // namespace node
65 }  // namespace blockchain
66
67 class Contact;
68 class Identifier;
69 class PasswordPrompt;
70 class PaymentCode;
71 // }  // namespace v1
72 }  // namespace opentxs
73 // NOLINTEND(modernize-concat-nested-namespaces)
74
75 namespace opentxs::api::crypto
76 {
80 class OPENTXS_EXPORT Blockchain
81 {
82 public:
83     using Chain = opentxs::blockchain::Type;
84     using Key = opentxs::blockchain::crypto::Key;
85     using Style = opentxs::blockchain::crypto::AddressStyle;
86     using Subchain = opentxs::blockchain::crypto::Subchain;
87     using DecodedAddress =
88         std::tuple<OTData, Style, UnallocatedSet<Chain>, bool>;
89     using ContactList = UnallocatedSet<OTIdentifier>;
90     using Txid = opentxs::blockchain::block::Txid;
91     using TxidHex = UnallocatedCString;
92     using PatternID = opentxs::blockchain::PatternID;
93     using AccountData = std::pair<Chain, OTNymID>;
94
95     // Throws std::out_of_range for invalid chains
96     static auto Bip44(Chain chain) noexcept(false) -> Bip44Type;
97     static auto Bip44Path(
98         Chain chain,
99         const UnallocatedCString& seed,
100         AllocateOutput destination) noexcept(false) -> bool;
101
103     virtual auto Account(const identifier::Nym& nymID, const Chain chain) const
104         noexcept(false) -> const opentxs::blockchain::crypto::Account& = 0;
105     virtual auto AccountList(const identifier::Nym& nymID) const noexcept
106         -> UnallocatedSet<OTIdentifier> = 0;
107     virtual auto AccountList(const Chain chain) const noexcept
108         -> UnallocatedSet<OTIdentifier> = 0;
109     virtual auto AccountList() const noexcept
110         -> UnallocatedSet<OTIdentifier> = 0;
111     virtual auto ActivityDescription(
112         const identifier::Nym& nym,
113         const Identifier& thread,
114         const UnallocatedCString& threadItemID) const noexcept
115         -> UnallocatedCString = 0;
116     virtual auto ActivityDescription(
117         const identifier::Nym& nym,
118         const Chain chain,
119         const opentxs::blockchain::block::bitcoin::Transaction& transaction)
120         const noexcept -> UnallocatedCString = 0;
121     virtual auto AssignContact(
122         const identifier::Nym& nymID,
123         const Identifier& accountID,
124         const Subchain subchain,
125         const Bip32Index index,
126         const Identifier& label) const noexcept -> bool = 0;
```

```
127     virtual auto AssignLabel(
128         const identifier::Nym& nymID,
129         const Identifier& accountID,
130         const Subchain subchain,
131         const Bip32Index index,
132         const UnallocatedCString& label) const noexcept -> bool = 0;
133     virtual auto AssignTransactionMemo(
134         const TxidHex& id,
135         const UnallocatedCString& label) const noexcept -> bool = 0;
136     virtual auto CalculateAddress(
137         const opentxs::blockchain::Type chain,
138         const opentxs::blockchain::crypto::AddressStyle format,
139         const Data& pubkey) const noexcept -> UnallocatedCString = 0;
140     virtual auto Confirm(
141         const Key key,
142         const opentxs::blockchain::block::Txid& tx) const noexcept -> bool = 0;
143     virtual auto DecodeAddress(const UnallocatedCString& encoded) const noexcept
144         -> DecodedAddress = 0;
145     virtual auto EncodeAddress(
146         const Style style,
147         const Chain chain,
148         const Data& data) const noexcept -> UnallocatedCString = 0;
150     virtual auto GetKey(const Key& id) const noexcept(false)
151         -> const opentxs::blockchain::crypto::Element& = 0;
153     virtual auto HDSubaccount(
154         const identifier::Nym& nymID,
155         const Identifier& accountID) const noexcept(false)
156         -> const opentxs::blockchain::crypto::HD& = 0;
157     virtual auto IndexItem(const ReadView bytes) const noexcept
158         -> PatternID = 0;
159     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
160         -> const crypto::internal::Blockchain& = 0;
161     virtual auto LoadTransactionBitcoin(const Txid& id) const noexcept
162         -> std::unique_ptr<
163             const opentxs::blockchain::block::bitcoin::Transaction> = 0;
164     virtual auto LoadTransactionBitcoin(const TxidHex& id) const noexcept
165         -> std::unique_ptr<
166             const opentxs::blockchain::block::bitcoin::Transaction> = 0;
167     virtual auto LookupAccount(const Identifier& id) const noexcept
168         -> AccountData = 0;
169     virtual auto LookupContacts(
170         const UnallocatedCString& address) const noexcept -> ContactList = 0;
171     virtual auto LookupContacts(const Data& pubkeyHash) const noexcept
172         -> ContactList = 0;
173     virtual auto NewHDSubaccount(
174         const identifier::Nym& nymID,
175         const opentxs::blockchain::crypto::HDProtocol standard,
176         const Chain chain,
177         const PasswordPrompt& reason) const noexcept -> OTIdentifier = 0;
178     virtual auto NewHDSubaccount(
179         const identifier::Nym& nymID,
180         const opentxs::blockchain::crypto::HDProtocol standard,
181         const Chain derivationChain,
182         const Chain targetChain,
183         const PasswordPrompt& reason) const noexcept -> OTIdentifier = 0;
184     virtual auto NewPaymentCodeSubaccount(
185         const identifier::Nym& nymID,
186         const opentxs::PaymentCode& local,
187         const opentxs::PaymentCode& remote,
188         const ReadView& view,
189         const Chain chain,
190         const PasswordPrompt& reason) const noexcept -> OTIdentifier = 0;
191     virtual auto Owner(const Identifier& accountID) const noexcept
192         -> const identifier::Nym& = 0;
193     virtual auto Owner(const Key& key) const noexcept
194         -> const identifier::Nym& = 0;
196     virtual auto PaymentCodeSubaccount(
197         const identifier::Nym& nymID,
198         const Identifier& accountID) const noexcept(false)
199         -> const opentxs::blockchain::crypto::PaymentCode& = 0;
200     virtual auto RecipientContact(const Key& key) const noexcept
201         -> OTIdentifier = 0;
202     virtual auto Release(const Key key) const noexcept -> bool = 0;
203     virtual auto SenderContact(const Key& key) const noexcept
204         -> OTIdentifier = 0;
205     virtual auto SubaccountList(const identifier::Nym& nymID, const Chain chain)
206         const noexcept -> UnallocatedSet<OTIdentifier> = 0;
207     virtual auto Unconfirm(
208         const Key key,
209         const opentxs::blockchain::block::Txid& tx,
210         const Time time = Clock::now()) const noexcept -> bool = 0;
212     virtual auto Wallet(const Chain chain) const noexcept(false)
213         -> const opentxs::blockchain::crypto::Wallet& = 0;
214
215     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
216         -> crypto::internal::Blockchain& = 0;
217
```

```
218     OPENTXS_NO_EXPORT virtual ~Blockchain() = default;
219
220 protected:
221     Blockchain() noexcept = default;
222
223 private:
224     Blockchain(const Blockchain&) = delete;
225     Blockchain(Blockchain&&) = delete;
226     auto operator=(const Blockchain&) -> Blockchain& = delete;
227     auto operator=(Blockchain&&) -> Blockchain& = delete;
228 };
229 }  // namespace opentxs::api::crypto
```

7.4 Blockchain.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/blockchain/Types.hpp"
11 #include "opentxs/util/Container.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace api
19 {
20 namespace network
21 {
22 namespace internal
23 {
24 class Blockchain;
25 }  // namespace internal
26 }  // namespace network
27 }  // namespace api
28
29 namespace blockchain
30 {
31 namespace node
32 {
33 class Manager;
34 }  // namespace node
35 }  // namespace blockchain
36 // }  // namespace v1
37 }  // namespace opentxs
38 // NOLINTEND(modernize-concat-nested-namespaces)
39
40 namespace opentxs::api::network
41 {
45 class OPENTXS_EXPORT Blockchain
46 {
47 public:
48     struct Imp;
49
50     using Chain = opentxs::blockchain::Type;
51     using Endpoints = UnallocatedVector<UnallocatedCString>;
52
53     auto AddSyncServer(const UnallocatedCString& endpoint) const noexcept
54         -> bool;
55     auto ConnectedSyncServers() const noexcept -> Endpoints;
56     auto DeleteSyncServer(const UnallocatedCString& endpoint) const noexcept
57         -> bool;
58     auto Disable(const Chain type) const noexcept -> bool;
59     auto Enable(const Chain type, const UnallocatedCString& seednode = "")
60         const noexcept -> bool;
61     auto EnabledChains() const noexcept -> UnallocatedSet<Chain>;
63     auto GetChain(const Chain type) const noexcept(false)
64         -> const opentxs::blockchain::node::Manager&;
65     auto GetSyncServers() const noexcept -> Endpoints;
66     OPENTXS_NO_EXPORT auto Internal() const noexcept -> internal::Blockchain&;
67     auto Start(const Chain type, const UnallocatedCString& seednode = "")
68         const noexcept -> bool;
69     auto StartSyncServer(
70         const UnallocatedCString& syncEndpoint,
71         const UnallocatedCString& publicSyncEndpoint,
72         const UnallocatedCString& updateEndpoint,
```

```
73          const UnallocatedCString& publicUpdateEndpoint) const noexcept -> bool;
74      auto Stop(const Chain type) const noexcept -> bool;
75
76      OPENTXS_NO_EXPORT Blockchain(Imp* imp) noexcept;
77
78      OPENTXS_NO_EXPORT ~Blockchain();
79
80  private:
81      Imp* imp_;
82
83      Blockchain() = delete;
84      Blockchain(const Blockchain&) = delete;
85      Blockchain(Blockchain&&) = delete;
86      auto operator=(const Blockchain&) -> Blockchain& = delete;
87      auto operator=(Blockchain&&) -> Blockchain& = delete;
88  };
89  }  // namespace opentxs::api::network
```

7.5 Config.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  // IWYU pragma: no_include "opentxs/crypto/ParameterType.hpp"
7  // IWYU pragma: no_include "opentxs/crypto/key/asymmetric/Algorithm.hpp"
8  // IWYU pragma: no_include "opentxs/crypto/key/symmetric/Algorithm.hpp"
9
10 #pragma once
11
12 #include "opentxs/Version.hpp"  // IWYU pragma: associated
13
14 #include <cstdint>
15
16 #include "opentxs/crypto/Types.hpp"
17 #include "opentxs/crypto/key/Types.hpp"
18
19 // NOLINTBEGIN(modernize-concat-nested-namespaces)
20 namespace opentxs  // NOLINT
21 {
22 // inline namespace v1
23 // {
24 namespace api
25 {
26 namespace crypto
27 {
28 namespace internal
29 {
30 class Config;
31 }  // namespace internal
32 }  // namespace crypto
33 }  // namespace api
34 // }  // namespace v1
35 }  // namespace opentxs
36 // NOLINTEND(modernize-concat-nested-namespaces)
37
38 namespace opentxs::api::crypto
39 {
40 auto HaveHDKeys() noexcept -> bool;
41 auto HaveSupport(opentxs::crypto::ParameterType) noexcept -> bool;
42 auto HaveSupport(opentxs::crypto::key::asymmetric::Algorithm) noexcept -> bool;
43 auto HaveSupport(opentxs::crypto::key::symmetric::Algorithm) noexcept -> bool;
44
48 class OPENTXS_EXPORT Config
49 {
50 public:
51     OPENTXS_NO_EXPORT virtual auto InternalConfig() const noexcept
52         -> const internal::Config& = 0;
53     virtual auto IterationCount() const -> std::uint32_t = 0;
54     virtual auto SymmetricSaltSize() const -> std::uint32_t = 0;
55     virtual auto SymmetricKeySize() const -> std::uint32_t = 0;
56     virtual auto SymmetricKeySizeMax() const -> std::uint32_t = 0;
57     virtual auto SymmetricIvSize() const -> std::uint32_t = 0;
58     virtual auto SymmetricBufferSize() const -> std::uint32_t = 0;
59     virtual auto PublicKeysize() const -> std::uint32_t = 0;
60     virtual auto PublicKeysizeMax() const -> std::uint32_t = 0;
61
62     OPENTXS_NO_EXPORT virtual auto InternalConfig() noexcept
63         -> internal::Config& = 0;
64
65     OPENTXS_NO_EXPORT virtual ~Config() = default;
66
```

```
67 protected:
68     Config() = default;
69
70 private:
71     Config(const Config&) = delete;
72     Config(Config&&) = delete;
73     auto operator=(const Config&) -> Config& = delete;
74     auto operator=(Config&&) -> Config& = delete;
75 };
76 }  // namespace opentxs::api::crypto
```

7.6 Crypto.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 // NOLINTBEGIN(modernize-concat-nested-namespaces)
11 namespace opentxs  // NOLINT
12 {
13 // inline namespace v1
14 // {
15 namespace api
16 {
17 namespace crypto
18 {
19 class Config;
20 class Encode;
21 class Hash;
22 class Util;
23 }  // namespace crypto
24
25 namespace internal
26 {
27 class Crypto;
28 }  // namespace internal
29 }  // namespace api
30
31 namespace crypto
32 {
33 class Bip32;
34 class Bip39;
35 }  // namespace crypto
36 // }  // namespace v1
37 }  // namespace opentxs
38 // NOLINTEND(modernize-concat-nested-namespaces)
39
40 namespace opentxs::api
41 {
46 class OPENTXS_EXPORT Crypto
47 {
48 public:
49     virtual auto BIP32() const noexcept -> const opentxs::crypto::Bip32& = 0;
50     virtual auto BIP39() const noexcept -> const opentxs::crypto::Bip39& = 0;
52     virtual auto Config() const noexcept -> const crypto::Config& = 0;
53     virtual auto Encode() const noexcept -> const crypto::Encode& = 0;
54     virtual auto Hash() const noexcept -> const crypto::Hash& = 0;
55     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
56         -> const internal::Crypto& = 0;
58     virtual auto Util() const noexcept -> const crypto::Util& = 0;
59
60     OPENTXS_NO_EXPORT virtual auto Internal() noexcept -> internal::Crypto& = 0;
61
62     OPENTXS_NO_EXPORT virtual ~Crypto() = default;
63
64 protected:
65     Crypto() = default;
66
67 private:
68     Crypto(const Crypto&) = delete;
69     Crypto(Crypto&&) = delete;
70     auto operator=(const Crypto&) -> Crypto& = delete;
71     auto operator=(Crypto&&) -> Crypto& = delete;
72 };
73 }  // namespace opentxs::api
```

7.7 Crypto.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/api/crypto/Crypto.hpp"
11
12 // NOLINTBEGIN(modernize-concat-nested-namespaces)
13 namespace opentxs  // NOLINT
14 {
15 // inline namespace v1
16 // {
17 namespace api
18 {
19 namespace crypto
20 {
21 class Asymmetric;
22 class Blockchain;
23 class Seed;
24 class Symmetric;
25 }  // namespace crypto
26
27 namespace session
28 {
29 namespace internal
30 {
31 class Crypto;
32 }  // namespace internal
33 }  // namespace session
34 }  // namespace api
35 // }  // namespace v1
36 }  // namespace opentxs
37 // NOLINTEND(modernize-concat-nested-namespaces)
38
39 namespace opentxs::api::session
40 {
41 class OPENTXS_EXPORT Crypto : virtual public api::Crypto
42 {
43 public:
44     virtual auto Asymmetric() const noexcept -> const crypto::Asymmetric& = 0;
45     virtual auto Blockchain() const noexcept -> const crypto::Blockchain& = 0;
46     OPENTXS_NO_EXPORT virtual auto InternalSession() const noexcept
47         -> const internal::Crypto& = 0;
48     virtual auto Seed() const noexcept -> const crypto::Seed& = 0;
49     virtual auto Symmetric() const noexcept -> const crypto::Symmetric& = 0;
50
51     OPENTXS_NO_EXPORT virtual auto InternalSession() noexcept
52         -> internal::Crypto& = 0;
53
54     OPENTXS_NO_EXPORT ~Crypto() override = default;
55
56 protected:
57     Crypto() = default;
58
59 private:
60     Crypto(const Crypto&) = delete;
61     Crypto(Crypto&&) = delete;
62     auto operator=(const Crypto&) -> Crypto& = delete;
63     auto operator=(Crypto&&) -> Crypto& = delete;
64 };
65 }  // namespace opentxs::api::session
```

7.8 Encode.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstdint>
11
12 #include "opentxs/core/Data.hpp"
13 #include "opentxs/core/String.hpp"
```

```
14 #include "opentxs/util/Container.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace api
22 {
23 namespace crypto
24 {
25 namespace internal
26 {
27 class Encode;
28 }  // namespace internal
29 }  // namespace crypto
30 }  // namespace api
31 // }  // namespace v1
32 }  // namespace opentxs
33 // NOLINTEND(modernize-concat-nested-namespaces)
34
35 namespace opentxs::api::crypto
36 {
40 class Encode
41 {
42 public:
43     virtual auto DataEncode(const UnallocatedCString& input) const
44         -> UnallocatedCString = 0;
45     virtual auto DataEncode(const Data& input) const -> UnallocatedCString = 0;
46     virtual auto DataDecode(const UnallocatedCString& input) const
47         -> UnallocatedCString = 0;
48     virtual auto IdentifierEncode(const Data& input) const
49         -> UnallocatedCString = 0;
50     virtual auto IdentifierDecode(const UnallocatedCString& input) const
51         -> UnallocatedCString = 0;
52     OPENTXS_NO_EXPORT virtual auto InternalEncode() const noexcept
53         -> const internal::Encode& = 0;
54     virtual auto IsBase62(const UnallocatedCString& str) const -> bool = 0;
55     virtual auto Nonce(const std::uint32_t size) const -> OTString = 0;
56     virtual auto Nonce(const std::uint32_t size, Data& rawOutput) const
57         -> OTString = 0;
58     virtual auto RandomFilename() const -> UnallocatedCString = 0;
59     virtual auto SanatizeBase58(const UnallocatedCString& input) const
60         -> UnallocatedCString = 0;
61     virtual auto SanatizeBase64(const UnallocatedCString& input) const
62         -> UnallocatedCString = 0;
63     virtual auto Z85Encode(const Data& input) const -> UnallocatedCString = 0;
64     virtual auto Z85Encode(const UnallocatedCString& input) const
65         -> UnallocatedCString = 0;
66     virtual auto Z85Decode(const Data& input) const -> OTData = 0;
67     virtual auto Z85Decode(const UnallocatedCString& input) const
68         -> UnallocatedCString = 0;
69
70     OPENTXS_NO_EXPORT virtual auto InternalEncode() noexcept
71         -> internal::Encode& = 0;
72
73     OPENTXS_NO_EXPORT virtual ~Encode() = default;
74
75 protected:
76     Encode() = default;
77
78 private:
79     Encode(const Encode&) = delete;
80     Encode(Encode&&) = delete;
81     auto operator=(const Encode&) -> Encode& = delete;
82     auto operator=(Encode&&) -> Encode& = delete;
83 };
84 }  // namespace opentxs::api::crypto
```

7.9 Hash.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstdint>
11
12 #include "opentxs/crypto/Types.hpp"
```

```
13 #include "opentxs/util/Bytes.hpp"
14 #include "opentxs/util/Container.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace api
22 {
23 namespace crypto
24 {
25 namespace internal
26 {
27 class Hash;
28 }  // namespace internal
29 }  // namespace crypto
30 }  // namespace api
31
32 namespace network
33 {
34 namespace zeromq
35 {
36 class Frame;
37 }  // namespace zeromq
38 }  // namespace network
39
40 class Data;
41 class Secret;
42 // }  // namespace v1
43 }  // namespace opentxs
44 // NOLINTEND(modernize-concat-nested-namespaces)
45
46 namespace opentxs::api::crypto
47 {
51 class OPENTXS_EXPORT Hash
52 {
53 public:
54     virtual auto Digest(
55         const opentxs::crypto::HashType hashType,
56         const ReadView data,
57         const AllocateOutput destination) const noexcept -> bool = 0;
58     virtual auto Digest(
59         const opentxs::crypto::HashType hashType,
60         const opentxs::network::zeromq::Frame& data,
61         const AllocateOutput destination) const noexcept -> bool = 0;
62     virtual auto Digest(
63         const std::uint32_t type,
64         const ReadView data,
65         const AllocateOutput encodedDestination) const noexcept -> bool = 0;
66     virtual auto HMAC(
67         const opentxs::crypto::HashType hashType,
68         const ReadView key,
69         const ReadView& data,
70         const AllocateOutput digest) const noexcept -> bool = 0;
71     OPENTXS_NO_EXPORT virtual auto InternalHash() const noexcept
72         -> const internal::Hash& = 0;
73     virtual auto MurmurHash3_32(
74         const std::uint32_t& key,
75         const Data& data,
76         std::uint32_t& output) const noexcept -> void = 0;
77     virtual auto PKCS5_PBKDF2_HMAC(
78         const Data& input,
79         const Data& salt,
80         const std::size_t iterations,
81         const opentxs::crypto::HashType hashType,
82         const std::size_t bytes,
83         Data& output) const noexcept -> bool = 0;
84     virtual auto PKCS5_PBKDF2_HMAC(
85         const Secret& input,
86         const Data& salt,
87         const std::size_t iterations,
88         const opentxs::crypto::HashType hashType,
89         const std::size_t bytes,
90         Data& output) const noexcept -> bool = 0;
91     virtual auto PKCS5_PBKDF2_HMAC(
92         const UnallocatedCString& input,
93         const Data& salt,
94         const std::size_t iterations,
95         const opentxs::crypto::HashType hashType,
96         const std::size_t bytes,
97         Data& output) const noexcept -> bool = 0;
98     virtual auto Scrypt(
99         const ReadView input,
100         const ReadView salt,
101         const std::uint64_t N,
102         const std::uint32_t r,
```

```
103         const std::uint32_t p,
104         const std::size_t bytes,
105         AllocateOutput writer) const noexcept -> bool = 0;
106
107     OPENTXS_NO_EXPORT virtual auto InternalHash() noexcept
108         -> internal::Hash& = 0;
109
110     OPENTXS_NO_EXPORT virtual ~Hash() = default;
111
112 protected:
113     Hash() noexcept = default;
114
115 private:
116     Hash(const Hash&) = delete;
117     Hash(Hash&&) = delete;
118     auto operator=(const Hash&) -> Hash& = delete;
119     auto operator=(Hash&&) -> Hash& = delete;
120 };
121 }  // namespace opentxs::api::crypto
```

7.10 Seed.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/crypto/key/asymmetric/Role.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include <cstdint>
13 #include <memory>
14 #include <string_view>
15 #include <tuple>
16
17 #include "opentxs/core/Secret.hpp"
18 #include "opentxs/crypto/Types.hpp"
19 #include "opentxs/crypto/key/Symmetric.hpp"
20 #include "opentxs/crypto/key/Types.hpp"
21 #include "opentxs/util/Container.hpp"
22 #include "opentxs/util/Numbers.hpp"
23
24 // NOLINTBEGIN(modernize-concat-nested-namespaces)
25 namespace opentxs  // NOLINT
26 {
27 // inline namespace v1
28 // {
29 namespace api
30 {
31 namespace crypto
32 {
33 namespace internal
34 {
35 class Seed;
36 }  // namespace internal
37 }  // namespace crypto
38 }  // namespace api
39
40 namespace crypto
41 {
42 namespace key
43 {
44 class HD;
45 class Secp256k1;
46 }  // namespace key
47
48 class Seed;
49 }  // namespace crypto
50
51 class Identifier;
52 class PasswordPrompt;
53 class Secret;
54 // }  // namespace v1
55 }  // namespace opentxs
56 // NOLINTEND(modernize-concat-nested-namespaces)
57
58 namespace opentxs::api::crypto
59 {
63 class OPENTXS_EXPORT Seed
64 {
```

```
65 public:
66     virtual auto AccountChildKey(
67         const ReadView& path,
68         const BIP44Chain internal,
69         const Bip32Index index,
70         const PasswordPrompt& reason) const
71         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
72     virtual auto AllowedSeedTypes() const noexcept -> const
73         UnallocatedMap<opentxs::crypto::SeedStyle, UnallocatedCString>& = 0;
74     virtual auto AllowedLanguages(
75         const opentxs::crypto::SeedStyle type) const noexcept -> const
76         UnallocatedMap<opentxs::crypto::Language, UnallocatedCString>& = 0;
77     virtual auto AllowedSeedStrength(
78         const opentxs::crypto::SeedStyle type) const noexcept -> const
79         UnallocatedMap<opentxs::crypto::SeedStrength, UnallocatedCString>& = 0;
80     virtual auto Bip32Root(
81         const UnallocatedCString& seedID,
82         const PasswordPrompt& reason) const -> UnallocatedCString = 0;
83     virtual auto DefaultSeed() const
84         -> std::pair<UnallocatedCString, std::size_t> = 0;
85     virtual auto GetHDKey(
86         const UnallocatedCString& seedID,
87         const opentxs::crypto::EcdsaCurve& curve,
88         const UnallocatedVector<Bip32Index>& path,
89         const PasswordPrompt& reason) const
90         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
91     virtual auto GetHDKey(
92         const UnallocatedCString& seedID,
93         const opentxs::crypto::EcdsaCurve& curve,
94         const UnallocatedVector<Bip32Index>& path,
95         const opentxs::crypto::key::asymmetric::Role,
96         const PasswordPrompt& reason) const
97         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
98     virtual auto GetHDKey(
99         const UnallocatedCString& seedID,
100         const opentxs::crypto::EcdsaCurve& curve,
101         const UnallocatedVector<Bip32Index>& path,
102         const VersionNumber version,
103         const PasswordPrompt& reason) const
104         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
105     virtual auto GetHDKey(
106         const UnallocatedCString& seedID,
107         const opentxs::crypto::EcdsaCurve& curve,
108         const UnallocatedVector<Bip32Index>& path,
109         const opentxs::crypto::key::asymmetric::Role,
110         const VersionNumber version,
111         const PasswordPrompt& reason) const
112         -> std::unique_ptr<opentxs::crypto::key::HD> = 0;
113     virtual auto GetPaymentCode(
114         const UnallocatedCString& seedID,
115         const Bip32Index nym,
116         const std::uint8_t version,
117         const PasswordPrompt& reason) const
118         -> std::unique_ptr<opentxs::crypto::key::Secp256k1> = 0;
119     virtual auto GetStorageKey(
120         const UnallocatedCString& seedID,
121         const PasswordPrompt& reason) const -> OTSymmetricKey = 0;
122     virtual auto GetSeed(
123         const UnallocatedCString& seedID,
124         Bip32Index& index,
125         const PasswordPrompt& reason) const -> OTSecret = 0;
126     virtual auto GetSeed(const Identifier& id, const PasswordPrompt& reason)
127         const noexcept -> opentxs::crypto::Seed = 0;
128     virtual auto ImportRaw(const Secret& entropy, const PasswordPrompt& reason)
129         const -> UnallocatedCString = 0;
130     virtual auto ImportSeed(
131         const Secret& words,
132         const Secret& passphrase,
133         const opentxs::crypto::SeedStyle type,
134         const opentxs::crypto::Language lang,
135         const PasswordPrompt& reason,
136         const std::string_view comment = {}) const -> UnallocatedCString = 0;
137     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
138         -> const internal::Seed& = 0;
139     virtual auto LongestWord(
140         const opentxs::crypto::SeedStyle type,
141         const opentxs::crypto::Language lang) const noexcept -> std::size_t = 0;
142     virtual auto NewSeed(
143         const opentxs::crypto::SeedStyle type,
144         const opentxs::crypto::Language lang,
145         const opentxs::crypto::SeedStrength strength,
146         const PasswordPrompt& reason,
147         const std::string_view comment = {}) const -> UnallocatedCString = 0;
148     virtual auto Passphrase(
149         const UnallocatedCString& seedID,
150         const PasswordPrompt& reason) const -> UnallocatedCString = 0;
151     virtual auto SeedDescription(UnallocatedCString seedID) const noexcept
```

```
152         -> UnallocatedCString = 0;
153     virtual auto SetDefault(const Identifier& id) const noexcept -> bool = 0;
154     virtual auto SetSeedComment(
155         const Identifier& id,
156         const std::string_view comment) const noexcept -> bool = 0;
157     virtual auto ValidateWord(
158         const opentxs::crypto::SeedStyle type,
159         const opentxs::crypto::Language lang,
160         const std::string_view word) const noexcept
161         -> UnallocatedVector<UnallocatedCString> = 0;
162     virtual auto WordCount(
163         const opentxs::crypto::SeedStyle type,
164         const opentxs::crypto::SeedStrength strength) const noexcept
165         -> std::size_t = 0;
166     virtual auto Words(
167         const UnallocatedCString& seedID,
168         const PasswordPrompt& reason) const -> UnallocatedCString = 0;
169
170     OPENTXS_NO_EXPORT virtual auto Internal() noexcept -> internal::Seed& = 0;
171
172     OPENTXS_NO_EXPORT virtual ~Seed() = default;
173
174 protected:
175     Seed() = default;
176
177 private:
178     Seed(const Seed&) = delete;
179     Seed(Seed&&) = delete;
180     auto operator=(const Seed&) -> Seed& = delete;
181     auto operator=(Seed&&) -> Seed& = delete;
182 };
183 }  // namespace opentxs::api::crypto
```

7.11 Symmetric.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstdint>
11
12 #include "opentxs/crypto/key/Symmetric.hpp"
13 #include "opentxs/crypto/key/symmetric/Algorithm.hpp"
14 #include "opentxs/crypto/key/symmetric/Source.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace api
22 {
23 namespace crypto
24 {
25 namespace internal
26 {
27 class Symmetric;
28 }  // namespace internal
29 }  // namespace crypto
30 }  // namespace api
31
32 class Secret;
33 // }  // namespace v1
34 }  // namespace opentxs
35 // NOLINTEND(modernize-concat-nested-namespaces)
36
37 namespace opentxs::api::crypto
38 {
42 class OPENTXS_EXPORT Symmetric
43 {
44 public:
45     OPENTXS_NO_EXPORT virtual auto InternalSymmetric() const noexcept
46         -> const internal::Symmetric& = 0;
47     virtual auto IvSize(const opentxs::crypto::key::symmetric::Algorithm mode)
48         const -> std::size_t = 0;
49     virtual auto Key(
50         const PasswordPrompt& password,
51         const opentxs::crypto::key::symmetric::Algorithm mode =
```

```
52              opentxs::crypto::key::symmetric::Algorithm::ChaCha20Poly1305) const
53          -> OTSymmetricKey = 0;
54      virtual auto Key(
55          const ReadView& serializedCiphertext,
56          const opentxs::crypto::key::symmetric::Algorithm mode) const
57          -> OTSymmetricKey = 0;
58      virtual auto Key(
59          const Secret& seed,
60          const std::uint64_t operations = 0,
61          const std::uint64_t difficulty = 0,
62          const opentxs::crypto::key::symmetric::Algorithm mode =
63              opentxs::crypto::key::symmetric::Algorithm::ChaCha20Poly1305,
64          const opentxs::crypto::key::symmetric::Source type =
65              opentxs::crypto::key::symmetric::Source::Argon2i) const
66          -> OTSymmetricKey = 0;
67      virtual auto Key(
68          const Secret& seed,
69          const ReadView salt,
70          const std::uint64_t operations,
71          const std::uint64_t difficulty,
72          const std::uint64_t parallel,
73          const std::size_t bytes,
74          const opentxs::crypto::key::symmetric::Source type) const
75          -> OTSymmetricKey = 0;
76
77      OPENTXS_NO_EXPORT virtual auto InternalSymmetric() noexcept
78          -> internal::Symmetric& = 0;
79
80      OPENTXS_NO_EXPORT virtual ~Symmetric() = default;
81
82  protected:
83      Symmetric() = default;
84
85  private:
86      Symmetric(const Symmetric&) = delete;
87      Symmetric(Symmetric&&) = delete;
88      auto operator=(const Symmetric&) -> Symmetric& = delete;
89      auto operator=(Symmetric&&) -> Symmetric& = delete;
90  };
91  }  // namespace opentxs::api::crypto
```

7.12 Util.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstdint>
11
12 // NOLINTBEGIN(modernize-concat-nested-namespaces)
13 namespace opentxs  // NOLINT
14 {
15 // inline namespace v1
16 // {
17 namespace api
18 {
19 namespace crypto
20 {
21 namespace internal
22 {
23 class Util;
24 }  // namespace internal
25 }  // namespace crypto
26 }  // namespace api
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::api::crypto
32 {
36 class Util
37 {
38 public:
39      OPENTXS_NO_EXPORT virtual auto InternalUtil() const noexcept
40          -> const internal::Util& = 0;
41      virtual auto RandomizeMemory(void* destination, const std::size_t size)
42          const -> bool = 0;
43
```

```
44     OPENTXS_NO_EXPORT virtual auto InternalUtil() noexcept
45         -> internal::Util& = 0;
46
47     OPENTXS_NO_EXPORT virtual ~Util() = default;
48
49 protected:
50     Util() = default;
51
52 private:
53     Util(const Util&) = delete;
54     Util(Util&&) = delete;
55     auto operator=(const Util&) -> Util& = delete;
56     auto operator=(Util&&) -> Util& = delete;
57 };
58 }  // namespace opentxs::api::crypto
```

7.13 Factory.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/core/Secret.hpp"
11
12 // NOLINTBEGIN(modernize-concat-nested-namespaces)
13 namespace opentxs  // NOLINT
14 {
15 // inline namespace v1
16 // {
17 namespace api
18 {
19 namespace internal
20 {
21 class Factory;
22 }  // namespace internal
23 }  // namespace api
24 // }  // namespace v1
25 }  // namespace opentxs
26 // NOLINTEND(modernize-concat-nested-namespaces)
27
28 namespace opentxs::api
29 {
37 class OPENTXS_EXPORT Factory
38 {
39 public:
40     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
41         -> const internal::Factory& = 0;
42     virtual auto Secret(const std::size_t bytes) const noexcept -> OTSecret = 0;
43     virtual auto SecretFromBytes(const ReadView bytes) const noexcept
44         -> OTSecret = 0;
45     virtual auto SecretFromText(const std::string_view text) const noexcept
46         -> OTSecret = 0;
47
48     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
49         -> internal::Factory& = 0;
50
51     OPENTXS_NO_EXPORT virtual ~Factory() = default;
52
53 protected:
54     Factory() noexcept = default;
55
56 private:
57     Factory(const Factory&) = delete;
58     Factory(Factory&&) = delete;
59     auto operator=(const Factory&) -> Factory& = delete;
60     auto operator=(Factory&&) -> Factory& = delete;
61 };
62 }  // namespace opentxs::api
```

7.14 Factory.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
```

```
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/otx/blind/CashType.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include <cstdint>
13
14 #include "opentxs/api/Factory.hpp"
15 #include "opentxs/blockchain/Types.hpp"
16 #include "opentxs/blockchain/block/Types.hpp"
17 #include "opentxs/blockchain/crypto/Types.hpp"
18 #include "opentxs/blockchain/p2p/Address.hpp"
19 #include "opentxs/core/Armored.hpp"
20 #include "opentxs/core/Data.hpp"
21 #include "opentxs/core/String.hpp"
22 #include "opentxs/core/Types.hpp"
23 #include "opentxs/core/contract/BasketContract.hpp"
24 #include "opentxs/core/contract/CurrencyContract.hpp"
25 #include "opentxs/core/contract/SecurityContract.hpp"
26 #include "opentxs/core/contract/ServerContract.hpp"
27 #include "opentxs/core/contract/Unit.hpp"
28 #include "opentxs/core/contract/peer/BailmentNotice.hpp"
29 #include "opentxs/core/contract/peer/BailmentReply.hpp"
30 #include "opentxs/core/contract/peer/BailmentRequest.hpp"
31 #include "opentxs/core/contract/peer/ConnectionReply.hpp"
32 #include "opentxs/core/contract/peer/ConnectionRequest.hpp"
33 #include "opentxs/core/contract/peer/NoticeAcknowledgement.hpp"
34 #include "opentxs/core/contract/peer/OutBailmentReply.hpp"
35 #include "opentxs/core/contract/peer/OutBailmentRequest.hpp"
36 #include "opentxs/core/contract/peer/PeerObject.hpp"
37 #include "opentxs/core/contract/peer/PeerReply.hpp"
38 #include "opentxs/core/contract/peer/PeerRequest.hpp"
39 #include "opentxs/core/contract/peer/StoreSecret.hpp"
40 #include "opentxs/core/contract/peer/Types.hpp"
41 #include "opentxs/core/identifier/Generic.hpp"
42 #include "opentxs/core/identifier/Notary.hpp"
43 #include "opentxs/core/identifier/Nym.hpp"
44 #include "opentxs/core/identifier/UnitDefinition.hpp"
45 #include "opentxs/crypto/Envelope.hpp"
46 #include "opentxs/crypto/Types.hpp"
47 #include "opentxs/crypto/key/Asymmetric.hpp"
48 #include "opentxs/crypto/key/Keypair.hpp"
49 #include "opentxs/crypto/key/Symmetric.hpp"
50 #include "opentxs/crypto/key/asymmetric/Role.hpp"      // TODO remove
51 #include "opentxs/crypto/key/symmetric/Algorithm.hpp"  // TODO remove
52 #include "opentxs/identity/wot/claim/Types.hpp"
53 #include "opentxs/otx/blind/Types.hpp"
54 #include "opentxs/util/Bytes.hpp"
55 #include "opentxs/util/Container.hpp"
56 #include "opentxs/util/Numbers.hpp"
57 #include "opentxs/util/PasswordPrompt.hpp"
58
59 // NOLINTBEGIN(modernize-concat-nested-namespaces)
60 namespace opentxs  // NOLINT
61 {
62 // inline namespace v1
63 // {
64 namespace api
65 {
66 namespace session
67 {
68 namespace internal
69 {
70 class Factory;
71 }  // namespace internal
72 }  // namespace session
73
74 class Session;
75 }  // namespace api
76
77 namespace blockchain
78 {
79 namespace block
80 {
81 namespace bitcoin
82 {
83 class Block;
84 class Script;
85 class Transaction;
86 }  // namespace bitcoin
87
88 class Block;
89 class Hash;
90 class Header;
```

```
91 }  // namespace block
92 }  // namespace blockchain
93
94 namespace crypto
95 {
96 class SymmetricProvider;
97 }  // namespace crypto
98
99 namespace display
100 {
101 class Definition;
102 }  // namespace display
103
104 namespace identifier
105 {
106 class Nym;
107 class Notary;
108 class UnitDefinition;
109 }  // namespace identifier
110
111 namespace network
112 {
113 namespace p2p
114 {
115 class Base;
116 }  // namespace p2p
117
118 namespace zeromq
119 {
120 class Frame;
121 class Message;
122 }  // namespace zeromq
123 }  // namespace network
124
125 namespace otx
126 {
127 namespace blind
128 {
129 class Mint;
130 class Purse;
131 }  // namespace blind
132
133 namespace context
134 {
135 class Server;
136 }  // namespace context
137 }  // namespace otx
138
139 class Secret;
140 class PaymentCode;
141 // }  // namespace v1
142 }  // namespace opentxs
143 // NOLINTEND(modernize-concat-nested-namespaces)
144
145 namespace opentxs::api::session
146 {
150 class OPENTXS_EXPORT Factory : virtual public api::Factory
151 {
152 public:
153     virtual auto Armored() const -> OTArmored = 0;
154     virtual auto Armored(const UnallocatedCString& input) const
155         -> OTArmored = 0;
156     virtual auto Armored(const opentxs::Data& input) const -> OTArmored = 0;
157     virtual auto Armored(const opentxs::String& input) const -> OTArmored = 0;
158     virtual auto Armored(const opentxs::crypto::Envelope& input) const
159         -> OTArmored = 0;
160     virtual auto AsymmetricKey(
161         const opentxs::crypto::Parameters& params,
162         const opentxs::PasswordPrompt& reason,
163         const opentxs::crypto::key::asymmetric::Role role =
164             opentxs::crypto::key::asymmetric::Role::Sign,
165         const VersionNumber version =
166             opentxs::crypto::key::Asymmetric::DefaultVersion) const
167         -> OTAsymmetricKey = 0;
168     virtual auto BailmentNotice(
169         const Nym_p& nym,
170         const identifier::Nym& recipientID,
171         const identifier::UnitDefinition& unitID,
172         const identifier::Notary& serverID,
173         const Identifier& requestID,
174         const UnallocatedCString& txid,
175         const Amount& amount,
176         const opentxs::PasswordPrompt& reason) const noexcept(false)
177         -> OTBailmentNotice = 0;
178     virtual auto BailmentReply(
179         const Nym_p& nym,
180         const identifier::Nym& initiator,
```

```
181         const Identifier& request,
182         const identifier::Notary& server,
183         const UnallocatedCString& terms,
184         const opentxs::PasswordPrompt& reason) const noexcept(false)
185         -> OTBailmentReply = 0;
186     virtual auto BailmentRequest(
187         const Nym_p& nym,
188         const identifier::Nym& recipient,
189         const identifier::UnitDefinition& unit,
190         const identifier::Notary& server,
191         const opentxs::PasswordPrompt& reason) const noexcept(false)
192         -> OTBailmentRequest = 0;
193     virtual auto BailmentRequest(const Nym_p& nym, const ReadView& view) const
194         noexcept(false) -> OTBailmentRequest = 0;
195     virtual auto BasketContract(
196         const Nym_p& nym,
197         const UnallocatedCString& shortname,
198         const UnallocatedCString& terms,
199         const std::uint64_t weight,
200         const UnitType unitOfAccount,
201         const VersionNumber version,
202         const display::Definition& displayDefinition,
203         const Amount& redemptionIncrement) const noexcept(false)
204         -> OTBasketContract = 0;
205 #if OT_BLOCKCHAIN
206     virtual auto BitcoinBlock(
207         const opentxs::blockchain::Type chain,
208         const ReadView bytes) const noexcept
209         -> std::shared_ptr<
210             const opentxs::blockchain::block::bitcoin::Block> = 0;
211     using Transaction_p =
212         std::shared_ptr<const opentxs::blockchain::block::bitcoin::Transaction>;
213     using AbortFunction = std::function<bool()>;
214     virtual auto BitcoinBlock(
215         const opentxs::blockchain::block::Header& previous,
216         const Transaction_p generationTransaction,
217         const std::uint32_t nBits,
218         const UnallocatedVector<Transaction_p>& extraTransactions = {},
219         const std::int32_t version = 2,
220         const AbortFunction abort = {}) const noexcept
221         -> std::shared_ptr<
222             const opentxs::blockchain::block::bitcoin::Block> = 0;
223     using OutputBuilder = std::tuple<
224         opentxs::blockchain::Amount,
225         std::unique_ptr<const opentxs::blockchain::block::bitcoin::Script>,
226         UnallocatedSet<opentxs::blockchain::crypto::Key>>;
227     virtual auto BitcoinGenerationTransaction(
228         const opentxs::blockchain::Type chain,
229         const opentxs::blockchain::block::Height height,
230         UnallocatedVector<OutputBuilder>&& outputs,
231         const UnallocatedCString& coinbase = {},
232         const std::int32_t version = 1) const noexcept -> Transaction_p = 0;
233     virtual auto BitcoinScriptNullData(
234         const opentxs::blockchain::Type chain,
235         const UnallocatedVector<ReadView>& data) const noexcept
236         -> std::unique_ptr<
237             const opentxs::blockchain::block::bitcoin::Script> = 0;
238     virtual auto BitcoinScriptP2MS(
239         const opentxs::blockchain::Type chain,
240         const std::uint8_t M,
241         const std::uint8_t N,
242         const UnallocatedVector<const opentxs::crypto::key::EllipticCurve*>&
243             publicKeys) const noexcept
244         -> std::unique_ptr<
245             const opentxs::blockchain::block::bitcoin::Script> = 0;
246     virtual auto BitcoinScriptP2PK(
247         const opentxs::blockchain::Type chain,
248         const opentxs::crypto::key::EllipticCurve& publicKey) const noexcept
249         -> std::unique_ptr<
250             const opentxs::blockchain::block::bitcoin::Script> = 0;
251     virtual auto BitcoinScriptP2PKH(
252         const opentxs::blockchain::Type chain,
253         const opentxs::crypto::key::EllipticCurve& publicKey) const noexcept
254         -> std::unique_ptr<
255             const opentxs::blockchain::block::bitcoin::Script> = 0;
256     virtual auto BitcoinScriptP2SH(
257         const opentxs::blockchain::Type chain,
258         const opentxs::blockchain::block::bitcoin::Script& script)
259         const noexcept -> std::unique_ptr<
260             const opentxs::blockchain::block::bitcoin::Script> = 0;
261     virtual auto BitcoinScriptP2WPKH(
262         const opentxs::blockchain::Type chain,
263         const opentxs::crypto::key::EllipticCurve& publicKey) const noexcept
264         -> std::unique_ptr<
265             const opentxs::blockchain::block::bitcoin::Script> = 0;
266     virtual auto BitcoinScriptP2WSH(
267         const opentxs::blockchain::Type chain,
```

```
268         const opentxs::blockchain::block::bitcoin::Script& script)
269         const noexcept -> std::unique_ptr<
270             const opentxs::blockchain::block::bitcoin::Script> = 0;
271     virtual auto BitcoinTransaction(
272         const opentxs::blockchain::Type chain,
273         const ReadView bytes,
274         const bool isGeneration,
275         const Time& time = Clock::now()) const noexcept
276         -> std::unique_ptr<
277             const opentxs::blockchain::block::bitcoin::Transaction> = 0;
278     virtual auto BlockchainAddress(
279         const opentxs::blockchain::p2p::Protocol protocol,
280         const opentxs::blockchain::p2p::Network network,
281         const opentxs::Data& bytes,
282         const std::uint16_t port,
283         const opentxs::blockchain::Type chain,
284         const Time lastConnected,
285         const UnallocatedSet<opentxs::blockchain::p2p::Service>& services,
286         const bool incoming = false) const -> OTBlockchainAddress = 0;
287     virtual auto BlockchainAddress(
288         const opentxs::blockchain::p2p::Address::SerializedType& serialized)
289         const -> OTBlockchainAddress = 0;
290     virtual auto BlockchainSyncMessage(
291         const opentxs::network::zeromq::Message& in) const noexcept
292         -> std::unique_ptr<opentxs::network::p2p::Base> = 0;
293     using BlockHeaderP = std::unique_ptr<opentxs::blockchain::block::Header>;
294     virtual auto BlockHeader(const ReadView protobuf) const -> BlockHeaderP = 0;
295     virtual auto BlockHeader(
296         const opentxs::blockchain::Type type,
297         const ReadView native) const -> BlockHeaderP = 0;
298     virtual auto BlockHeader(const opentxs::blockchain::block::Block& block)
299         const -> BlockHeaderP = 0;
300     virtual auto BlockHeaderForUnitTests(
301         const opentxs::blockchain::block::Hash& hash,
302         const opentxs::blockchain::block::Hash& parent,
303         const opentxs::blockchain::block::Height height) const
304         -> BlockHeaderP = 0;
305 #endif  // OT_BLOCKCHAIN
306     virtual auto ConnectionReply(
307         const Nym_p& nym,
308         const identifier::Nym& initiator,
309         const Identifier& request,
310         const identifier::Notary& server,
311         const bool ack,
312         const UnallocatedCString& url,
313         const UnallocatedCString& login,
314         const UnallocatedCString& password,
315         const UnallocatedCString& key,
316         const opentxs::PasswordPrompt& reason) const noexcept(false)
317         -> OTConnectionReply = 0;
318     virtual auto ConnectionRequest(
319         const Nym_p& nym,
320         const identifier::Nym& recipient,
321         const contract::peer::ConnectionInfoType type,
322         const identifier::Notary& server,
323         const opentxs::PasswordPrompt& reason) const noexcept(false)
324         -> OTConnectionRequest = 0;
325     virtual auto CurrencyContract(
326         const Nym_p& nym,
327         const UnallocatedCString& shortname,
328         const UnallocatedCString& terms,
329         const UnitType unitOfAccount,
330         const VersionNumber version,
331         const opentxs::PasswordPrompt& reason,
332         const display::Definition& displayDefinition,
333         const Amount& redemptionIncrement) const noexcept(false)
334         -> OTCurrencyContract = 0;
335     virtual auto Data() const -> OTData = 0;
336     virtual auto Data(const opentxs::Armored& input) const -> OTData = 0;
337     virtual auto Data(const opentxs::network::zeromq::Frame& input) const
338         -> OTData = 0;
339     virtual auto Data(const std::uint8_t input) const -> OTData = 0;
340     virtual auto Data(const std::uint32_t input) const -> OTData = 0;
341     virtual auto Data(const UnallocatedVector<unsigned char>& input) const
342         -> OTData = 0;
343     virtual auto Data(const UnallocatedVector<std::byte>& input) const
344         -> OTData = 0;
345     virtual auto DataFromBytes(const ReadView input) const -> OTData = 0;
346     virtual auto DataFromHex(const ReadView input) const -> OTData = 0;
347     virtual auto Envelope() const noexcept -> OTEnvelope = 0;
348     virtual auto Envelope(const opentxs::Armored& ciphertext) const
349         noexcept(false) -> OTEnvelope = 0;
350     virtual auto Envelope(
351         const opentxs::crypto::Envelope::SerializedType& serialized) const
352         noexcept(false) -> OTEnvelope = 0;
353     virtual auto Envelope(const opentxs::ReadView& serialized) const
354         noexcept(false) -> OTEnvelope = 0;
```

```
355     virtual auto Identifier() const -> OTIdentifier = 0;
356     virtual auto Identifier(const UnallocatedCString& serialized) const
357         -> OTIdentifier = 0;
358     virtual auto Identifier(const opentxs::String& serialized) const
359         -> OTIdentifier = 0;
360     virtual auto Identifier(const opentxs::Contract& contract) const
361         -> OTIdentifier = 0;
362     virtual auto Identifier(const opentxs::Item& item) const
363         -> OTIdentifier = 0;
364     virtual auto Identifier(const ReadView bytes) const -> OTIdentifier = 0;
365     virtual auto Identifier(const opentxs::network::zeromq::Frame& bytes) const
366         -> OTIdentifier = 0;
367     OPENTXS_NO_EXPORT virtual auto InternalSession() const noexcept
368         -> const internal::Factory& = 0;
369     virtual auto Keypair(
370         const opentxs::crypto::Parameters& nymParameters,
371         const VersionNumber version,
372         const opentxs::crypto::key::asymmetric::Role role,
373         const opentxs::PasswordPrompt& reason) const -> OTKeypair = 0;
374     virtual auto Keypair(
375         const UnallocatedCString& fingerprint,
376         const Bip32Index nym,
377         const Bip32Index credset,
378         const Bip32Index credindex,
379         const opentxs::crypto::EcdsaCurve& curve,
380         const opentxs::crypto::key::asymmetric::Role role,
381         const opentxs::PasswordPrompt& reason) const -> OTKeypair = 0;
382     virtual auto Mint() const noexcept -> otx::blind::Mint = 0;
383     virtual auto Mint(const otx::blind::CashType type) const noexcept
384         -> otx::blind::Mint = 0;
385     virtual auto Mint(
386         const identifier::Notary& notary,
387         const identifier::UnitDefinition& unit) const noexcept
388         -> otx::blind::Mint = 0;
389     virtual auto Mint(
390         const otx::blind::CashType type,
391         const identifier::Notary& notary,
392         const identifier::UnitDefinition& unit) const noexcept
393         -> otx::blind::Mint = 0;
394     virtual auto Mint(
395         const identifier::Notary& notary,
396         const identifier::Nym& serverNym,
397         const identifier::UnitDefinition& unit) const noexcept
398         -> otx::blind::Mint = 0;
399     virtual auto Mint(
400         const otx::blind::CashType type,
401         const identifier::Notary& notary,
402         const identifier::Nym& serverNym,
403         const identifier::UnitDefinition& unit) const noexcept
404         -> otx::blind::Mint = 0;
405     virtual auto NymID() const -> OTNymID = 0;
406     virtual auto NymID(const UnallocatedCString& serialized) const
407         -> OTNymID = 0;
408     virtual auto NymID(const opentxs::String& serialized) const -> OTNymID = 0;
409     virtual auto NymID(const opentxs::network::zeromq::Frame& bytes) const
410         -> OTNymID = 0;
411     virtual auto NymIDFromPaymentCode(
412         const UnallocatedCString& serialized) const -> OTNymID = 0;
413     virtual auto OutbailmentReply(
414         const Nym_p& nym,
415         const identifier::Nym& initiator,
416         const opentxs::Identifier& request,
417         const identifier::Notary& server,
418         const UnallocatedCString& terms,
419         const opentxs::PasswordPrompt& reason) const noexcept(false)
420         -> OTOutbailmentReply = 0;
421     virtual auto OutbailmentRequest(
422         const Nym_p& nym,
423         const identifier::Nym& recipientID,
424         const identifier::UnitDefinition& unitID,
425         const identifier::Notary& serverID,
426         const Amount& amount,
427         const UnallocatedCString& terms,
428         const opentxs::PasswordPrompt& reason) const noexcept(false)
429         -> OTOutbailmentRequest = 0;
430     virtual auto PasswordPrompt(const UnallocatedCString& text) const
431         -> OTPasswordPrompt = 0;
432     virtual auto PasswordPrompt(const opentxs::PasswordPrompt& rhs) const
433         -> OTPasswordPrompt = 0;
434     virtual auto PaymentCode(const UnallocatedCString& base58) const noexcept
435         -> opentxs::PaymentCode = 0;
436     virtual auto PaymentCode(const ReadView& serialized) const noexcept
437         -> opentxs::PaymentCode = 0;
438     virtual auto PaymentCode(
439         const UnallocatedCString& seed,
440         const Bip32Index nym,
441         const std::uint8_t version,
```

```
442         const opentxs::PasswordPrompt& reason,
443         const bool bitmessage = false,
444         const std::uint8_t bitmessageVersion = 0,
445         const std::uint8_t bitmessageStream = 0) const noexcept
446         -> opentxs::PaymentCode = 0;
447     virtual auto PeerObject(
448         const Nym_p& senderNym,
449         const UnallocatedCString& message) const
450         -> std::unique_ptr<opentxs::PeerObject> = 0;
451     virtual auto PeerObject(
452         const Nym_p& senderNym,
453         const UnallocatedCString& payment,
454         const bool isPayment) const -> std::unique_ptr<opentxs::PeerObject> = 0;
455     virtual auto PeerObject(const Nym_p& senderNym, otx::blind::Purse&& purse)
456         const -> std::unique_ptr<opentxs::PeerObject> = 0;
457     virtual auto PeerObject(
458         const OTPeerRequest request,
459         const OTPeerReply reply,
460         const VersionNumber version) const
461         -> std::unique_ptr<opentxs::PeerObject> = 0;
462     virtual auto PeerObject(
463         const OTPeerRequest request,
464         const VersionNumber version) const
465         -> std::unique_ptr<opentxs::PeerObject> = 0;
466     virtual auto PeerObject(
467         const Nym_p& recipientNym,
468         const opentxs::Armored& encrypted,
469         const opentxs::PasswordPrompt& reason) const
470         -> std::unique_ptr<opentxs::PeerObject> = 0;
471     virtual auto PeerReply() const noexcept -> OTPeerReply = 0;
472     virtual auto PeerReply(const Nym_p& nym, const ReadView& view) const
473         noexcept(false) -> OTPeerReply = 0;
474     virtual auto PeerRequest() const noexcept -> OTPeerRequest = 0;
475     virtual auto PeerRequest(const Nym_p& nym, const ReadView& view) const
476         noexcept(false) -> OTPeerRequest = 0;
477     virtual auto Purse(
478         const otx::context::Server& context,
479         const identifier::UnitDefinition& unit,
480         const otx::blind::Mint& mint,
481         const Amount& totalValue,
482         const opentxs::PasswordPrompt& reason) const noexcept
483         -> otx::blind::Purse = 0;
484     virtual auto Purse(
485         const otx::context::Server& context,
486         const identifier::UnitDefinition& unit,
487         const otx::blind::Mint& mint,
488         const Amount& totalValue,
489         const otx::blind::CashType type,
490         const opentxs::PasswordPrompt& reason) const noexcept
491         -> otx::blind::Purse = 0;
492     virtual auto Purse(
493         const identity::Nym& owner,
494         const identifier::Notary& server,
495         const identifier::UnitDefinition& unit,
496         const opentxs::PasswordPrompt& reason) const noexcept
497         -> otx::blind::Purse = 0;
498     virtual auto Purse(
499         const identity::Nym& owner,
500         const identifier::Notary& server,
501         const identifier::UnitDefinition& unit,
502         const otx::blind::CashType type,
503         const opentxs::PasswordPrompt& reason) const noexcept
504         -> otx::blind::Purse = 0;
505     virtual auto ReplyAcknowledgement(
506         const Nym_p& nym,
507         const identifier::Nym& initiator,
508         const opentxs::Identifier& request,
509         const identifier::Notary& server,
510         const contract::peer::PeerRequestType type,
511         const bool& ack,
512         const opentxs::PasswordPrompt& reason) const noexcept(false)
513         -> OTReplyAcknowledgement = 0;
514     virtual auto SecurityContract(
515         const Nym_p& nym,
516         const UnallocatedCString& shortname,
517         const UnallocatedCString& terms,
518         const UnitType unitOfAccount,
519         const VersionNumber version,
520         const opentxs::PasswordPrompt& reason,
521         const display::Definition& displayDefinition,
522         const Amount& redemptionIncrement) const noexcept(false)
523         -> OTSecurityContract = 0;
524     virtual auto ServerContract() const noexcept(false) -> OTServerContract = 0;
525     virtual auto ServerID() const -> OTNotaryID = 0;
526     virtual auto ServerID(const UnallocatedCString& serialized) const
527         -> OTNotaryID = 0;
528     virtual auto ServerID(const opentxs::String& serialized) const
```

```
529         -> OTNotaryID = 0;
530     virtual auto ServerID(const opentxs::network::zeromq::Frame& bytes) const
531         -> OTNotaryID = 0;
532     virtual auto StoreSecret(
533         const Nym_p& nym,
534         const identifier::Nym& recipientID,
535         const contract::peer::SecretType type,
536         const UnallocatedCString& primary,
537         const UnallocatedCString& secondary,
538         const identifier::Notary& server,
539         const opentxs::PasswordPrompt& reason) const noexcept(false)
540         -> OTStoreSecret = 0;
542     virtual auto SymmetricKey() const -> OTSymmetricKey = 0;
551     virtual auto SymmetricKey(
552         const opentxs::crypto::SymmetricProvider& engine,
553         const opentxs::PasswordPrompt& password,
554         const opentxs::crypto::key::symmetric::Algorithm mode =
555             opentxs::crypto::key::symmetric::Algorithm::Error) const
556         -> OTSymmetricKey = 0;
569     virtual auto SymmetricKey(
570         const opentxs::crypto::SymmetricProvider& engine,
571         const opentxs::Secret& seed,
572         const std::uint64_t operations,
573         const std::uint64_t difficulty,
574         const std::size_t size,
575         const opentxs::crypto::key::symmetric::Source type) const
576         -> OTSymmetricKey = 0;
577     virtual auto SymmetricKey(
578         const opentxs::crypto::SymmetricProvider& engine,
579         const opentxs::Secret& seed,
580         const ReadView salt,
581         const std::uint64_t operations,
582         const std::uint64_t difficulty,
583         const std::uint64_t parallel,
584         const std::size_t size,
585         const opentxs::crypto::key::symmetric::Source type) const
586         -> OTSymmetricKey = 0;
593     virtual auto SymmetricKey(
594         const opentxs::crypto::SymmetricProvider& engine,
595         const opentxs::Secret& raw,
596         const opentxs::PasswordPrompt& reason) const -> OTSymmetricKey = 0;
597     virtual auto UnitID() const -> OTUnitID = 0;
598     virtual auto UnitID(const UnallocatedCString& serialized) const
599         -> OTUnitID = 0;
600     virtual auto UnitID(const opentxs::String& serialized) const
601         -> OTUnitID = 0;
602     virtual auto UnitID(const opentxs::network::zeromq::Frame& bytes) const
603         -> OTUnitID = 0;
604     virtual auto UnitDefinition() const noexcept -> OTUnitDefinition = 0;
605
606     OPENTXS_NO_EXPORT virtual auto InternalSession() noexcept
607         -> internal::Factory& = 0;
608
609     OPENTXS_NO_EXPORT ~Factory() override = default;
610
611 protected:
612     Factory() = default;
613
614 private:
615     Factory(const Factory&) = delete;
616     Factory(Factory&&) = delete;
617     auto operator=(const Factory&) -> Factory& = delete;
618     auto operator=(Factory&&) -> Factory& = delete;
619 };
620 }  // namespace opentxs::api::session
```

7.15 Asio.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/network/asio/Endpoint.hpp"
7 // IWYU pragma: no_include "opentxs/network/asio/Socket.hpp"
8
9 #pragma once
10
11 #include "opentxs/Version.hpp"  // IWYU pragma: associated
12
13 #include <cstdint>
14 #include <functional>
15 #include <future>
```

```
16 #include <string_view>
17
18 #include "opentxs/core/Data.hpp"
19 #include "opentxs/util/Container.hpp"
20
21 // NOLINTBEGIN(modernize-concat-nested-namespaces)
22 namespace opentxs  // NOLINT
23 {
24 // inline namespace v1
25 // {
26 namespace api
27 {
28 namespace network
29 {
30 namespace internal
31 {
32 class Asio;
33 }  // namespace internal
34 }  // namespace network
35 }  // namespace api
36
37 namespace network
38 {
39 namespace asio
40 {
41 class Endpoint;
42 class Socket;
43 }  // namespace asio
44
45 namespace zeromq
46 {
47 class Context;
48 }  // namespace zeromq
49 }  // namespace network
50 // }  // namespace v1
51 }  // namespace opentxs
52 // NOLINTEND(modernize-concat-nested-namespaces)
53
54 namespace opentxs::api::network
55 {
59 class OPENTXS_EXPORT Asio
60 {
61 public:
62     using Endpoint = opentxs::network::asio::Endpoint;
63     using Socket = opentxs::network::asio::Socket;
64     using Resolved = UnallocatedVector<Endpoint>;
65     using AcceptCallback = std::function<void(Socket&&)>;
66
86     auto Accept(const Endpoint& endpoint, AcceptCallback cb) const noexcept
87         -> bool;
88     auto Close(const Endpoint& endpoint) const noexcept -> bool;
89     auto GetPublicAddress4() const noexcept -> std::shared_future<OTData>;
90     auto GetPublicAddress6() const noexcept -> std::shared_future<OTData>;
91     OPENTXS_NO_EXPORT auto Internal() const noexcept -> internal::Asio&;
92
100     auto MakeSocket(const Endpoint& endpoint) const noexcept -> Socket;
101
114     auto NotificationEndpoint() const noexcept -> const char*;
115     auto Resolve(std::string_view server, std::uint16_t port) const noexcept
116         -> Resolved;
117
118     OPENTXS_NO_EXPORT auto Init() noexcept -> void;
119     OPENTXS_NO_EXPORT auto Shutdown() noexcept -> void;
120
121     OPENTXS_NO_EXPORT Asio(
122         const opentxs::network::zeromq::Context& zmq) noexcept;
123
124     OPENTXS_NO_EXPORT ~Asio();
125
126 private:
127     struct Imp;
128
129     Imp* imp_;
130
131     Asio() = delete;
132     Asio(const Asio&) = delete;
133     Asio(Asio&&) = delete;
134     auto operator=(const Asio&) -> Asio& = delete;
135     auto operator=(Asio&&) -> Asio& = delete;
136 };
137 }  // namespace opentxs::api::network
```

7.16 Dht.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/util/Container.hpp"
11
12 // NOLINTBEGIN(modernize-concat-nested-namespaces)
13 namespace opentxs  // NOLINT
14 {
15 // inline namespace v1
16 // {
17 namespace api
18 {
19 namespace network
20 {
21 namespace internal
22 {
23 class Dht;
24 }  // namespace internal
25 }  // namespace network
26 }  // namespace api
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::api::network
32 {
37 class OPENTXS_EXPORT Dht
38 {
39 public:
40     virtual auto GetPublicNym(const UnallocatedCString& key) const noexcept
41         -> void = 0;
42     virtual auto GetServerContract(const UnallocatedCString& key) const noexcept
43         -> void = 0;
44     virtual auto GetUnitDefinition(const UnallocatedCString& key) const noexcept
45         -> void = 0;
46     virtual auto Insert(
47         const UnallocatedCString& key,
48         const UnallocatedCString& value) const noexcept -> void = 0;
49     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
50         -> const internal::Dht& = 0;
51
52     OPENTXS_NO_EXPORT virtual auto Internal() noexcept -> internal::Dht& = 0;
53
54     OPENTXS_NO_EXPORT virtual ~Dht() = default;
55
56 protected:
57     Dht() = default;
58
59 private:
60     Dht(const Dht&) = delete;
61     Dht(Dht&&) = delete;
62     auto operator=(const Dht&) -> Dht& = delete;
63     auto operator=(Dht&&) -> Dht& = delete;
64 };
65 }  // namespace opentxs::api::network
```

7.17 Network.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 // NOLINTBEGIN(modernize-concat-nested-namespaces)
11 namespace opentxs  // NOLINT
12 {
13 // inline namespace v1
14 // {
15 namespace api
16 {
17 namespace network
```

```
18 {
19 class Asio;
20 class Blockchain;
21 class Dht;
22 }  // namespace network
23 }  // namespace api
24
25 namespace network
26 {
27 namespace zeromq
28 {
29 class Context;
30 }  // namespace zeromq
31 }  // namespace network
32 // }  // namespace v1
33 }  // namespace opentxs
34 // NOLINTEND(modernize-concat-nested-namespaces)
35
36 namespace opentxs::api::network
37 {
42 class OPENTXS_EXPORT Network
43 {
44 public:
45     struct Imp;
46
47     auto Asio() const noexcept -> const network::Asio&;
48     auto Blockchain() const noexcept -> const network::Blockchain&;
49     auto DHT() const noexcept -> const network::Dht&;
50     auto ZeroMQ() const noexcept -> const opentxs::network::zeromq::Context&;
51
52     OPENTXS_NO_EXPORT auto Shutdown() noexcept -> void;
53
54     OPENTXS_NO_EXPORT Network(Imp*) noexcept;
55
56     OPENTXS_NO_EXPORT ~Network();
57
58 private:
59     Imp* imp_;
60
61     Network() = delete;
62     Network(const Network&) = delete;
63     Network(Network&&) = delete;
64     auto operator=(const Network&) -> Network& = delete;
65     auto operator=(Network&&) -> Network& = delete;
66 };
67 }  // namespace opentxs::api::network
```

7.18 ZAP.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/network/zeromq/zap/Callback.hpp"
11 #include "opentxs/util/Container.hpp"
12
13 namespace opentxs::api::network
14 {
18 class OPENTXS_EXPORT ZAP
19 {
20 public:
21     using Callback = opentxs::network::zeromq::zap::Callback::ReceiveCallback;
22     using Policy = opentxs::network::zeromq::zap::Callback::Policy;
23
34     virtual auto RegisterDomain(
35         const UnallocatedCString& domain,
36         const Callback& callback) const -> bool = 0;
37
45     virtual auto SetDefaultPolicy(const Policy policy) const -> bool = 0;
46
47     OPENTXS_NO_EXPORT virtual ~ZAP() = default;
48
49 protected:
50     ZAP() = default;
51
52 private:
53     ZAP(const ZAP&) = delete;
54     ZAP(ZAP&&) = delete;
```

```
55     auto operator=(const ZAP&) -> ZAP& = delete;
56     auto operator=(ZAP&&) -> ZAP& = delete;
57 };
58 }  // namespace opentxs::api::network
```

7.19 ZMQ.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <memory>
12
13 #include "opentxs/core/Types.hpp"
14 #include "opentxs/network/Types.hpp"
15 #include "opentxs/util/Container.hpp"
16
17 // NOLINTBEGIN(modernize-concat-nested-namespaces)
18 namespace opentxs  // NOLINT
19 {
20 // inline namespace v1
21 // {
22 namespace network
23 {
24 namespace zeromq
25 {
26 class Context;
27 }  // namespace zeromq
28
29 class ServerConnection;
30 }  // namespace network
31
32 class Flag;
33 // }  // namespace v1
34 }  // namespace opentxs
35 // NOLINTEND(modernize-concat-nested-namespaces)
36
37 namespace opentxs::api::network
38 {
42 class OPENTXS_EXPORT ZMQ
43 {
44 public:
45     virtual auto Context() const
46         -> const opentxs::network::zeromq::Context& = 0;
47     virtual auto DefaultAddressType() const -> AddressType = 0;
48     virtual auto KeepAlive() const -> std::chrono::seconds = 0;
49     virtual void KeepAlive(const std::chrono::seconds duration) const = 0;
50     virtual auto Linger() const -> std::chrono::seconds = 0;
51     virtual auto ReceiveTimeout() const -> std::chrono::seconds = 0;
52     virtual auto Running() const -> const Flag& = 0;
53     virtual void RefreshConfig() const = 0;
54     virtual auto SendTimeout() const -> std::chrono::seconds = 0;
55     virtual auto Server(const UnallocatedCString& id) const
56         -> opentxs::network::ServerConnection& = 0;
57     virtual auto SetSocksProxy(const UnallocatedCString& proxy) const
58         -> bool = 0;
59     virtual auto SocksProxy() const -> UnallocatedCString = 0;
60     virtual auto SocksProxy(UnallocatedCString& proxy) const -> bool = 0;
61     virtual auto Status(const UnallocatedCString& server) const
62         -> opentxs::network::ConnectionState = 0;
63
64     OPENTXS_NO_EXPORT virtual ~ZMQ() = default;
65
66 protected:
67     ZMQ() = default;
68
69 private:
70     ZMQ(const ZMQ&) = delete;
71     ZMQ(ZMQ&&) = delete;
72     auto operator=(const ZMQ&) -> ZMQ& = delete;
73     auto operator=(const ZMQ&&) -> ZMQ& = delete;
74 };
75 }  // namespace opentxs::api::network
```

7.20 Periodic.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <functional>
11
12 #include "opentxs/util/Time.hpp"
13
14 namespace opentxs
15 {
16 using PeriodicTask = std::function<void()>;
17 }  // namespace opentxs
18
19 namespace opentxs::api
20 {
24 class OPENTXS_EXPORT Periodic
25 {
26 public:
28     virtual auto Cancel(const int task) const -> bool = 0;
30     virtual auto Reschedule(
31         const int task,
32         const std::chrono::seconds& interval) const -> bool = 0;
38     virtual auto Schedule(
39         const std::chrono::seconds& interval,
40         const opentxs::PeriodicTask& task,
41         const std::chrono::seconds& last = 0s) const -> int = 0;
42
43     OPENTXS_NO_EXPORT virtual ~Periodic() = default;
44
45 protected:
46     Periodic() = default;
47
48 private:
49     Periodic(const Periodic&) = delete;
50     Periodic(Periodic&&) = delete;
51     auto operator=(const Periodic&) -> Periodic& = delete;
52     auto operator=(Periodic&&) -> Periodic& = delete;
53 };
54 }  // namespace opentxs::api
```

7.21 Activity.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <cstddef>
12 #include <future>
13 #include <memory>
14 #include <tuple>
15 #include <utility>
16
17 #include "opentxs/core/contract/Unit.hpp"
18 #include "opentxs/otx/client/Types.hpp"
19 #include "opentxs/util/Bytes.hpp"
20 #include "opentxs/util/Container.hpp"
21 #include "opentxs/util/Time.hpp"
22 #include "opentxs/util/Types.hpp"
23
24 // NOLINTBEGIN(modernize-concat-nested-namespaces)
25 namespace opentxs  // NOLINT
26 {
27 // inline namespace v1
28 // {
29 namespace api
30 {
31 namespace session
32 {
33 namespace internal
34 {
```

```
35 class Activity;
36 class Session;
37 }  // namespace internal
38 }  // namespace session
39
40 class Session;
41 }  // namespace api
42
43 namespace blockchain
44 {
45 namespace block
46 {
47 namespace bitcoin
48 {
49 class Transaction;
50 }  // namespace bitcoin
51 }  // namespace block
52 }  // namespace blockchain
53
54 namespace identifier
55 {
56 class Nym;
57 }  // namespace identifier
58
59 namespace proto
60 {
61 class StorageThread;
62 }  // namespace proto
63
64 class Identifier;
65 class PasswordPrompt;
66 class PeerObject;
67 // }  // namespace v1
68 }  // namespace opentxs
69 // NOLINTEND(modernize-concat-nested-namespaces)
70
71 namespace opentxs::api::session
72 {
73 class OPENTXS_EXPORT Activity
74 {
75 public:
76     virtual auto AddBlockchainTransaction(
77         const blockchain::block::bitcoin::Transaction& transaction)
78         const noexcept -> bool = 0;
79     virtual auto AddPaymentEvent(
80         const identifier::Nym& nymID,
81         const Identifier& threadID,
82         const otx::client::StorageBox type,
83         const Identifier& itemID,
84         const Identifier& workflowID,
85         Time time) const noexcept -> bool = 0;
86     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
87         -> const internal::Activity& = 0;
93     virtual auto Mail(
94         const identifier::Nym& nym,
95         const otx::client::StorageBox box) const noexcept -> ObjectList = 0;
104     virtual auto MailRemove(
105         const identifier::Nym& nym,
106         const Identifier& id,
107         const otx::client::StorageBox box) const noexcept -> bool = 0;
116     virtual auto MailText(
117         const identifier::Nym& nym,
118         const Identifier& id,
119         const otx::client::StorageBox& box,
120         const PasswordPrompt& reason) const noexcept
121         -> std::shared_future<UnallocatedCString> = 0;
129     virtual auto MarkRead(
130         const identifier::Nym& nymId,
131         const Identifier& threadId,
132         const Identifier& itemId) const noexcept -> bool = 0;
140     virtual auto MarkUnread(
141         const identifier::Nym& nymId,
142         const Identifier& threadId,
143         const Identifier& itemId) const noexcept -> bool = 0;
152     virtual auto PaymentText(
153         const identifier::Nym& nym,
154         const UnallocatedCString& id,
155         const UnallocatedCString& workflow) const noexcept
156         -> std::shared_ptr<const UnallocatedCString> = 0;
162     virtual auto PreloadActivity(
163         const identifier::Nym& nymID,
164         const std::size_t count,
165         const PasswordPrompt& reason) const noexcept -> void = 0;
173     virtual auto PreloadThread(
174         const identifier::Nym& nymID,
175         const Identifier& threadID,
176         const std::size_t start,
```

```
177         const std::size_t count,
178         const PasswordPrompt& reason) const noexcept -> void = 0;
179     OPENTXS_NO_EXPORT virtual auto Thread(
180         const identifier::Nym& nymID,
181         const Identifier& threadID,
182         proto::StorageThread& serialized) const noexcept -> bool = 0;
183     virtual auto Thread(
184         const identifier::Nym& nymID,
185         const Identifier& threadID,
186         AllocateOutput output) const noexcept -> bool = 0;
192     virtual auto Threads(
193         const identifier::Nym& nym,
194         const bool unreadOnly = false) const noexcept -> ObjectList = 0;
199     virtual auto UnreadCount(const identifier::Nym& nym) const noexcept
200         -> std::size_t = 0;
208     virtual auto ThreadPublisher(const identifier::Nym& nym) const noexcept
209         -> UnallocatedCString = 0;
210
211     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
212         -> internal::Activity& = 0;
213
214     OPENTXS_NO_EXPORT virtual ~Activity() = default;
215
216 protected:
217     Activity() = default;
218
219 private:
220     Activity(const Activity&) = delete;
221     Activity(Activity&&) = delete;
222     auto operator=(const Activity&) -> Activity& = delete;
223     auto operator=(Activity&&) -> Activity& = delete;
224 };
225 }  // namespace opentxs::api::session
```

7.22 Client.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/api/session/Session.hpp"
11 #include "opentxs/util/Container.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace api
19 {
20 namespace network
21 {
22 class ZMQ;
23 }  // namespace network
24
25 namespace session
26 {
27 namespace internal
28 {
29 class Client;
30 }  // namespace internal
31
32 class Activity;
33 class Contacts;
34 class OTX;
35 class UI;
36 class Workflow;
37 }  // namespace session
38 }  // namespace api
39 // }  // namespace v1
40 }  // namespace opentxs
41 // NOLINTEND(modernize-concat-nested-namespaces)
42
43 namespace opentxs::api::session
44 {
48 class OPENTXS_EXPORT Client : virtual public api::Session
49 {
50 public:
```

```
52      virtual auto Activity() const -> const session::Activity& = 0;
54      virtual auto Contacts() const -> const api::session::Contacts& = 0;
55      OPENTXS_NO_EXPORT virtual auto InternalClient() const noexcept
56          -> const internal::Client& = 0;
58      virtual auto OTX() const -> const session::OTX& = 0;
60      virtual auto UI() const -> const session::UI& = 0;
62      virtual auto Workflow() const -> const session::Workflow& = 0;
64      virtual auto ZMQ() const -> const network::ZMQ& = 0;
65
66      OPENTXS_NO_EXPORT virtual auto InternalClient() noexcept
67          -> internal::Client& = 0;
68
69      OPENTXS_NO_EXPORT ~Client() override = default;
70
71  protected:
72      Client() = default;
73
74  private:
75      Client(const Client&) = delete;
76      Client(Client&&) = delete;
77      auto operator=(const Client&) -> Client& = delete;
78      auto operator=(Client&&) -> Client& = delete;
79  };
80  }  // namespace opentxs::api::session
```

7.23 Contacts.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  // IWYU pragma: no_include "opentxs/blockchain/BlockchainType.hpp"
7  // IWYU pragma: no_include "opentxs/blockchain/ClaimType.hpp"
8
9  #pragma once
10
11 #include "opentxs/Version.hpp"  // IWYU pragma: associated
12
13 #include <memory>
14
15 #include "opentxs/blockchain/Types.hpp"
16 #include "opentxs/core/Types.hpp"
17 #include "opentxs/core/identifier/Generic.hpp"
18 #include "opentxs/util/Types.hpp"
19
20 // NOLINTBEGIN(modernize-concat-nested-namespaces)
21 namespace opentxs  // NOLINT
22 {
23 // inline namespace v1
24 // {
25 namespace api
26 {
27 namespace session
28 {
29 namespace internal
30 {
31 class Contacts;
32 }  // namespace internal
33 }  // namespace session
34 }  // namespace api
35
36 namespace identifier
37 {
38 class Nym;
39 }  // namespace identifier
40
41 class Contact;
42 class Identifier;
43 class PaymentCode;
44 // }  // namespace v1
45 }  // namespace opentxs
46 // NOLINTEND(modernize-concat-nested-namespaces)
47
48 namespace opentxs::api::session
49 {
50 class OPENTXS_EXPORT Contacts
51 {
52 public:
53      virtual auto Contact(const Identifier& id) const
54          -> std::shared_ptr<const opentxs::Contact> = 0;
56      virtual auto ContactID(const identifier::Nym& nymID) const
57          -> OTIdentifier = 0;
```

```
58      virtual auto ContactList() const -> ObjectList = 0;
59      virtual auto ContactName(const Identifier& contactID) const
60          -> UnallocatedCString = 0;
61      virtual auto ContactName(const Identifier& contactID, UnitType currencyHint)
62          const -> UnallocatedCString = 0;
63      OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
64          -> const internal::Contacts& = 0;
65      virtual auto Merge(const Identifier& parent, const Identifier& child) const
66          -> std::shared_ptr<const opentxs::Contact> = 0;
67      virtual auto NewContact(const UnallocatedCString& label) const
68          -> std::shared_ptr<const opentxs::Contact> = 0;
69      virtual auto NewContact(
70          const UnallocatedCString& label,
71          const identifier::Nym& nymID,
72          const PaymentCode& paymentCode) const
73          -> std::shared_ptr<const opentxs::Contact> = 0;
74      virtual auto NewContactFromAddress(
75          const UnallocatedCString& address,
76          const UnallocatedCString& label,
77          const opentxs::blockchain::Type currency) const
78          -> std::shared_ptr<const opentxs::Contact> = 0;
79      virtual auto NymToContact(const identifier::Nym& nymID) const
80          -> OTIdentifier = 0;
81      virtual auto PaymentCodeToContact(
82          const PaymentCode& code,
83          const opentxs::blockchain::Type currency) const -> OTIdentifier = 0;
84      virtual auto PaymentCodeToContact(
85          const UnallocatedCString& code,
86          const opentxs::blockchain::Type currency) const -> OTIdentifier = 0;
87
88      OPENTXS_NO_EXPORT virtual auto Internal() noexcept
89          -> internal::Contacts& = 0;
90
91      OPENTXS_NO_EXPORT virtual ~Contacts() = default;
92
93  protected:
94      Contacts() = default;
95
96  private:
97      Contacts(const Contacts&) = delete;
98      Contacts(Contacts&&) = delete;
99      auto operator=(const Contacts&) -> Contacts& = delete;
100     auto operator=(Contacts&&) -> Contacts& = delete;
101 };
102 }  // namespace opentxs::api::session
```

7.24 Endpoints.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <string_view>
11
12 #include "opentxs/blockchain/Types.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace api
20 {
21 namespace session
22 {
23 namespace internal
24 {
25 class Endpoints;
26 }  // namespace internal
27 }  // namespace session
28 }  // namespace api
29
30 namespace identifier
31 {
32 class Nym;
33 }  // namespace identifier
34 // }  // namespace v1
35 }  // namespace opentxs
```

```
36 // NOLINTEND(modernize-concat-nested-namespaces)
37
38 namespace opentxs::api::session
39 {
40 class OPENTXS_EXPORT Endpoints
41 {
42 public:
52     virtual auto AccountUpdate() const noexcept -> std::string_view = 0;
53
63     virtual auto BlockchainAccountCreated() const noexcept
64         -> std::string_view = 0;
65
75     virtual auto BlockchainBalance() const noexcept -> std::string_view = 0;
76
86     virtual auto BlockchainBlockAvailable() const noexcept
87         -> std::string_view = 0;
88
98     virtual auto BlockchainBlockDownloadQueue() const noexcept
99         -> std::string_view = 0;
100
110     virtual auto BlockchainMempool() const noexcept -> std::string_view = 0;
111
121     virtual auto BlockchainNewFilter() const noexcept -> std::string_view = 0;
122
132     virtual auto BlockchainPeer() const noexcept -> std::string_view = 0;
133
143     virtual auto BlockchainPeerConnection() const noexcept
144         -> std::string_view = 0;
145
155     virtual auto BlockchainReorg() const noexcept -> std::string_view = 0;
156
166     virtual auto BlockchainScanProgress() const noexcept
167         -> std::string_view = 0;
168
178     virtual auto BlockchainStateChange() const noexcept -> std::string_view = 0;
179
189     virtual auto BlockchainSyncProgress() const noexcept
190         -> std::string_view = 0;
191
201     virtual auto BlockchainSyncServerUpdated() const noexcept
202         -> std::string_view = 0;
203
213     virtual auto BlockchainTransactions() const noexcept
214         -> std::string_view = 0;
215
225     virtual auto BlockchainTransactions(
226         const identifier::Nym& nym) const noexcept -> std::string_view = 0;
227
237     virtual auto BlockchainWalletUpdated() const noexcept
238         -> std::string_view = 0;
239
249     virtual auto ConnectionStatus() const noexcept -> std::string_view = 0;
250
260     virtual auto ContactUpdate() const noexcept -> std::string_view = 0;
261
271     virtual auto DhtRequestNym() const noexcept -> std::string_view = 0;
272
282     virtual auto DhtRequestServer() const noexcept -> std::string_view = 0;
283
293     virtual auto DhtRequestUnit() const noexcept -> std::string_view = 0;
294
304     virtual auto FindNym() const noexcept -> std::string_view = 0;
305
315     virtual auto FindServer() const noexcept -> std::string_view = 0;
316
326     virtual auto FindUnitDefinition() const noexcept -> std::string_view = 0;
327
328     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
329         -> const session::internal::Endpoints& = 0;
330
340     virtual auto IssuerUpdate() const noexcept -> std::string_view = 0;
341
351     virtual auto Messagability() const noexcept -> std::string_view = 0;
352
362     virtual auto MessageLoaded() const noexcept -> std::string_view = 0;
363
373     virtual auto NymCreated() const noexcept -> std::string_view = 0;
374
384     virtual auto NymDownload() const noexcept -> std::string_view = 0;
385
396     virtual auto PairEvent() const noexcept -> std::string_view = 0;
397
409     virtual auto PeerReplyUpdate() const noexcept -> std::string_view = 0;
410
422     virtual auto PeerRequestUpdate() const noexcept -> std::string_view = 0;
423
434     virtual auto PendingBailment() const noexcept -> std::string_view = 0;
```

```
435
445     virtual auto SeedUpdated() const noexcept -> std::string_view = 0;
446
457     virtual auto ServerReplyReceived() const noexcept -> std::string_view = 0;
458
469     virtual auto ServerRequestSent() const noexcept -> std::string_view = 0;
470
480     virtual auto ServerUpdate() const noexcept -> std::string_view = 0;
481
491     virtual auto Shutdown() const noexcept -> std::string_view = 0;
492
502     virtual auto TaskComplete() const noexcept -> std::string_view = 0;
503
513     virtual auto ThreadUpdate(const std::string_view thread) const noexcept
514         -> std::string_view = 0;
515
525     virtual auto UnitUpdate() const noexcept -> std::string_view = 0;
526
536     virtual auto WidgetUpdate() const noexcept -> std::string_view = 0;
537
547     virtual auto WorkflowAccountUpdate() const noexcept -> std::string_view = 0;
548
549     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
550         -> session::internal::Endpoints& = 0;
551
552     OPENTXS_NO_EXPORT virtual ~Endpoints() = default;
553
554 protected:
555     Endpoints() = default;
556
557 private:
558     Endpoints(const Endpoints&) = delete;
559     Endpoints(Endpoints&&) = delete;
560     auto operator=(const Endpoints&) -> Endpoints& = delete;
561     auto operator=(Endpoints&&) -> Endpoints& = delete;
562 };
563 }  // namespace opentxs::api::session
```

7.25 Notary.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstdint>
11 #include <memory>
12
13 #include "opentxs/api/session/Session.hpp"
14 #include "opentxs/util/Container.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace api
22 {
23 namespace session
24 {
25 namespace internal
26 {
27 class Notary;
28 }  // namespace internal
29 }  // namespace session
30 }  // namespace api
31
32 namespace identifier
33 {
34 class Nym;
35 class Notary;
36 class UnitDefinition;
37 }  // namespace identifier
38
39 namespace otx
40 {
41 namespace blind
42 {
43 class Mint;
```

```
44 }  // namespace blind
45 }  // namespace otx
46
47 namespace server
48 {
49 class Server;
50 }  // namespace server
51
52 class Options;
53 // }  // namespace v1
54 }  // namespace opentxs
55 // NOLINTEND(modernize-concat-nested-namespaces)
56
57 namespace opentxs::api::session
58 {
59 class OPENTXS_EXPORT Notary : virtual public api::Session
60 {
61 public:
62     static auto DefaultMintKeyBytes() noexcept -> std::size_t;
63
65     virtual auto DropIncoming(const int count) const -> void = 0;
67     virtual auto DropOutgoing(const int count) const -> void = 0;
68     virtual auto GetAdminNym() const -> UnallocatedCString = 0;
69     virtual auto GetAdminPassword() const -> UnallocatedCString = 0;
70     virtual auto GetPrivateMint(
71         const identifier::UnitDefinition& unitid,
72         std::uint32_t series) const noexcept -> otx::blind::Mint& = 0;
73     virtual auto GetPublicMint(const identifier::UnitDefinition& unitID)
74         const noexcept -> otx::blind::Mint& = 0;
75     virtual auto GetUserName() const -> UnallocatedCString = 0;
76     virtual auto GetUserTerms() const -> UnallocatedCString = 0;
77     virtual auto ID() const -> const identifier::Notary& = 0;
78     OPENTXS_NO_EXPORT virtual auto InternalNotary() const noexcept
79         -> const session::internal::Notary& = 0;
80     virtual auto NymID() const -> const identifier::Nym& = 0;
81     virtual auto ScanMints() const -> void = 0;
82     virtual auto Server() const -> opentxs::server::Server& = 0;
83     virtual auto SetMintKeySize(const std::size_t size) const -> void = 0;
84     virtual auto UpdateMint(const identifier::UnitDefinition& unitID) const
85         -> void = 0;
86
87     OPENTXS_NO_EXPORT virtual auto InternalNotary() noexcept
88         -> session::internal::Notary& = 0;
89
90     OPENTXS_NO_EXPORT ~Notary() override = default;
91
92 protected:
93     Notary() = default;
94
95 private:
96     Notary(const Notary&) = delete;
97     Notary(Notary&&) = delete;
98     auto operator=(const Notary&) -> Notary& = delete;
99     auto operator=(Notary&&) -> Notary& = delete;
100 };
101 }  // namespace opentxs::api::session
```

7.26 OTX.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <cstdint>
12 #include <future>
13 #include <memory>
14 #include <tuple>
15
16 #include "opentxs/core/UnitType.hpp"
17 #include "opentxs/core/contract/peer/Types.hpp"
18 #include "opentxs/core/identifier/Notary.hpp"
19 #include "opentxs/otx/Types.hpp"
20 #include "opentxs/otx/client/Types.hpp"
21 #include "opentxs/util/Container.hpp"
22 #include "opentxs/util/Time.hpp"
23
24 #define OT_CHEQUE_DAYS 30
```

```
25 #define OT_CHEQUE_HOURS 24 * OT_CHEQUE_DAYS
26 #define DEFAULT_PROCESS_INBOX_ITEMS 5
27
28 // NOLINTBEGIN(modernize-concat-nested-namespaces)
29 namespace opentxs  // NOLINT
30 {
31 // inline namespace v1
32 // {
33 namespace api
34 {
35 namespace session
36 {
37 namespace internal
38 {
39 class OTX;
40 }  // namespace internal
41 }  // namespace session
42 }  // namespace api
43
44 namespace contract
45 {
46 class Server;
47 }  // namespace contract
48
49 namespace identifier
50 {
51 class Nym;
52 class Notary;
53 class UnitDefinition;
54 }  // namespace identifier
55
56 class Amount;
57 class OTPayment;
58 // }  // namespace v1
59 }  // namespace opentxs
60 // NOLINTEND(modernize-concat-nested-namespaces)
61
62 namespace opentxs::api::session
63 {
64 class OPENTXS_EXPORT OTX
65 {
66 public:
67     using TaskID = int;
68     using MessageID = OTIdentifier;
69     using Result = std::pair<otx::LastReplyStatus, std::shared_ptr<Message>>;
70     using Future = std::shared_future<Result>;
71     using BackgroundTask = std::pair<TaskID, Future>;
72     using Finished = std::shared_future<void>;
73
74     virtual auto AcknowledgeBailment(
75         const identifier::Nym& localNymID,
76         const identifier::Notary& serverID,
77         const identifier::Nym& targetNymID,
78         const Identifier& requestID,
79         const UnallocatedCString& instructions,
80         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
81     virtual auto AcknowledgeNotice(
82         const identifier::Nym& localNymID,
83         const identifier::Notary& serverID,
84         const identifier::Nym& recipientID,
85         const Identifier& requestID,
86         const bool ack,
87         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
88     virtual auto AcknowledgeOutbailment(
89         const identifier::Nym& localNymID,
90         const identifier::Notary& serverID,
91         const identifier::Nym& recipientID,
92         const Identifier& requestID,
93         const UnallocatedCString& details,
94         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
95     virtual auto AcknowledgeConnection(
96         const identifier::Nym& localNymID,
97         const identifier::Notary& serverID,
98         const identifier::Nym& recipientID,
99         const Identifier& requestID,
100         const bool ack,
101         const UnallocatedCString& url,
102         const UnallocatedCString& login,
103         const UnallocatedCString& password,
104         const UnallocatedCString& key,
105         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
106     virtual auto AutoProcessInboxEnabled() const -> bool = 0;
107     virtual auto CanDeposit(
108         const identifier::Nym& recipientNymID,
109         const OTPayment& payment) const -> otx::client::Depositability = 0;
110     virtual auto CanDeposit(
111         const identifier::Nym& recipientNymID,
```

```
112         const Identifier& accountID,
113         const OTPayment& payment) const -> otx::client::Depositability = 0;
114     virtual auto CanMessage(
115         const identifier::Nym& senderNymID,
116         const Identifier& recipientContactID,
117         const bool startIntroductionServer = true) const
118         -> otx::client::Messagability = 0;
119     virtual auto CheckTransactionNumbers(
120         const identifier::Nym& nym,
121         const identifier::Notary& serverID,
122         const std::size_t quantity) const -> bool = 0;
123     virtual auto ContextIdle(
124         const identifier::Nym& nym,
125         const identifier::Notary& server) const -> Finished = 0;
130     virtual auto DepositCheques(const identifier::Nym& nymID) const
131         -> std::size_t = 0;
138     virtual auto DepositCheques(
139         const identifier::Nym& nymID,
140         const UnallocatedSet<OTIdentifier>& chequeIDs) const -> std::size_t = 0;
141     virtual auto DepositPayment(
142         const identifier::Nym& recipientNymID,
143         const std::shared_ptr<const OTPayment>& payment) const
144         -> BackgroundTask = 0;
145     virtual auto DepositPayment(
146         const identifier::Nym& recipientNymID,
147         const Identifier& accountID,
148         const std::shared_ptr<const OTPayment>& payment) const
149         -> BackgroundTask = 0;
151     virtual void DisableAutoaccept() const = 0;
152     virtual auto DownloadMint(
153         const identifier::Nym& nym,
154         const identifier::Notary& server,
155         const identifier::UnitDefinition& unit) const -> BackgroundTask = 0;
156     virtual auto DownloadNym(
157         const identifier::Nym& localNymID,
158         const identifier::Notary& serverID,
159         const identifier::Nym& targetNymID) const -> BackgroundTask = 0;
160     virtual auto DownloadNymbox(
161         const identifier::Nym& localNymID,
162         const identifier::Notary& serverID) const -> BackgroundTask = 0;
163     virtual auto DownloadServerContract(
164         const identifier::Nym& localNymID,
165         const identifier::Notary& serverID,
166         const identifier::Notary& contractID) const -> BackgroundTask = 0;
167     virtual auto DownloadUnitDefinition(
168         const identifier::Nym& localNymID,
169         const identifier::Notary& serverID,
170         const identifier::UnitDefinition& contractID) const
171         -> BackgroundTask = 0;
172     virtual auto FindNym(const identifier::Nym& nymID) const
173         -> BackgroundTask = 0;
174     virtual auto FindNym(
175         const identifier::Nym& nymID,
176         const identifier::Notary& serverIDHint) const -> BackgroundTask = 0;
177     virtual auto FindServer(const identifier::Notary& serverID) const
178         -> BackgroundTask = 0;
179     virtual auto FindUnitDefinition(
180         const identifier::UnitDefinition& unit) const -> BackgroundTask = 0;
181     virtual auto InitiateBailment(
182         const identifier::Nym& localNymID,
183         const identifier::Notary& serverID,
184         const identifier::Nym& targetNymID,
185         const identifier::UnitDefinition& instrumentDefinitionID,
186         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
187     virtual auto InitiateOutbailment(
188         const identifier::Nym& localNymID,
189         const identifier::Notary& serverID,
190         const identifier::Nym& targetNymID,
191         const identifier::UnitDefinition& instrumentDefinitionID,
192         const Amount amount,
193         const UnallocatedCString& message,
194         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
195     virtual auto InitiateRequestConnection(
196         const identifier::Nym& localNymID,
197         const identifier::Notary& serverID,
198         const identifier::Nym& targetNymID,
199         const contract::peer::ConnectionInfoType& type,
200         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
201     virtual auto InitiateStoreSecret(
202         const identifier::Nym& localNymID,
203         const identifier::Notary& serverID,
204         const identifier::Nym& targetNymID,
205         const contract::peer::SecretType& type,
206         const UnallocatedCString& primary,
207         const UnallocatedCString& secondary,
208         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
209     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
```

```
210         -> const internal::OTX& = 0;
211     virtual auto IntroductionServer() const -> const identifier::Notary& = 0;
212     virtual auto IssueUnitDefinition(
213         const identifier::Nym& localNymID,
214         const identifier::Notary& serverID,
215         const identifier::UnitDefinition& unitID,
216         const UnitType advertise = UnitType::Error,
217         const UnallocatedCString& label = "") const -> BackgroundTask = 0;
218     virtual auto MessageContact(
219         const identifier::Nym& senderNymID,
220         const Identifier& contactID,
221         const UnallocatedCString& message,
222         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
223     virtual auto MessageStatus(const TaskID taskID) const
224         -> std::pair<otx::client::ThreadStatus, MessageID> = 0;
225     virtual auto NotifyBailment(
226         const identifier::Nym& localNymID,
227         const identifier::Notary& serverID,
228         const identifier::Nym& targetNymID,
229         const identifier::UnitDefinition& instrumentDefinitionID,
230         const Identifier& requestID,
231         const UnallocatedCString& txid,
232         const Amount amount,
233         const otx::client::SetID setID = {}) const -> BackgroundTask = 0;
234     virtual auto PayContact(
235         const identifier::Nym& senderNymID,
236         const Identifier& contactID,
237         std::shared_ptr<const OTPayment> payment) const -> BackgroundTask = 0;
238     virtual auto PayContactCash(
239         const identifier::Nym& senderNymID,
240         const Identifier& contactID,
241         const Identifier& workflowID) const -> BackgroundTask = 0;
242     virtual auto ProcessInbox(
243         const identifier::Nym& localNymID,
244         const identifier::Notary& serverID,
245         const Identifier& accountID) const -> BackgroundTask = 0;
246     virtual auto PublishServerContract(
247         const identifier::Nym& localNymID,
248         const identifier::Notary& serverID,
249         const Identifier& contractID) const -> BackgroundTask = 0;
250     virtual void Refresh() const = 0;
251     virtual auto RefreshCount() const -> std::uint64_t = 0;
252     virtual auto RegisterAccount(
253         const identifier::Nym& localNymID,
254         const identifier::Notary& serverID,
255         const identifier::UnitDefinition& unitID,
256         const UnallocatedCString& label = "") const -> BackgroundTask = 0;
257     virtual auto RegisterNym(
258         const identifier::Nym& localNymID,
259         const identifier::Notary& serverID,
260         const bool resync = false) const -> BackgroundTask = 0;
261     virtual auto RegisterNymPublic(
262         const identifier::Nym& nymID,
263         const identifier::Notary& server,
264         const bool setContactData,
265         const bool forcePrimary = false,
266         const bool resync = false) const -> BackgroundTask = 0;
267     virtual auto SetIntroductionServer(const contract::Server& contract) const
268         -> OTNotaryID = 0;
269     virtual auto SendCheque(
270         const identifier::Nym& localNymID,
271         const Identifier& sourceAccountID,
272         const Identifier& recipientContactID,
273         const Amount value,
274         const UnallocatedCString& memo,
275         const Time validFrom = Clock::now(),
276         const Time validTo =
277             (Clock::now() + std::chrono::hours(OT_CHEQUE_HOURS))) const
278         -> BackgroundTask = 0;
279     virtual auto SendExternalTransfer(
280         const identifier::Nym& localNymID,
281         const identifier::Notary& serverID,
282         const Identifier& sourceAccountID,
283         const Identifier& targetAccountID,
284         const Amount& value,
285         const UnallocatedCString& memo) const -> BackgroundTask = 0;
286     virtual auto SendTransfer(
287         const identifier::Nym& localNymID,
288         const identifier::Notary& serverID,
289         const Identifier& sourceAccountID,
290         const Identifier& targetAccountID,
291         const Amount& value,
292         const UnallocatedCString& memo) const -> BackgroundTask = 0;
293     virtual void StartIntroductionServer(
294         const identifier::Nym& localNymID) const = 0;
295     virtual auto Status(const TaskID taskID) const
296         -> otx::client::ThreadStatus = 0;
```

```
297     virtual auto WithdrawCash(
298         const identifier::Nym& nymID,
299         const identifier::Notary& serverID,
300         const Identifier& account,
301         const Amount value) const -> BackgroundTask = 0;
302
303     OPENTXS_NO_EXPORT virtual auto Internal() noexcept -> internal::OTX& = 0;
304
305     OPENTXS_NO_EXPORT virtual ~OTX() = default;
306
307 protected:
308     OTX() = default;
309
310 private:
311     OTX(const OTX&) = delete;
312     OTX(OTX&&) = delete;
313     auto operator=(const OTX&) -> OTX& = delete;
314     auto operator=(OTX&&) -> OTX& = delete;
315 };
316 }  // namespace opentxs::api::session
```

7.27 Session.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11
12 #include "opentxs/api/Periodic.hpp"
13 #include "opentxs/util/Container.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace api
21 {
22 namespace crypto
23 {
24 class Asymmetric;
25 class Seed;
26 class Symmetric;
27 }  // namespace crypto
28
29 namespace network
30 {
31 class Network;
32 }  // namespace network
33
34 namespace session
35 {
36 namespace internal
37 {
38 class Session;
39 }  // namespace internal
40
41 class Crypto;
42 class Endpoints;
43 class Factory;
44 class Storage;
45 class Wallet;
46 }  // namespace session
47
48 class Crypto;
49 class Settings;
50 }  // namespace api
51
52 class Options;
53 // }  // namespace v1
54 }  // namespace opentxs
55 // NOLINTEND(modernize-concat-nested-namespaces)
56
57 class QObject;
58
59 namespace opentxs::api
60 {
```

```
64 class OPENTXS_EXPORT Session : virtual public Periodic
65 {
66 public:
68     virtual auto Config() const noexcept -> const api::Settings& = 0;
70     virtual auto Crypto() const noexcept -> const session::Crypto& = 0;
72     virtual auto DataFolder() const noexcept -> const UnallocatedCString& = 0;
74     virtual auto Endpoints() const noexcept -> const session::Endpoints& = 0;
76     virtual auto Factory() const noexcept -> const session::Factory& = 0;
78     virtual auto GetOptions() const noexcept -> const Options& = 0;
79     virtual auto Instance() const noexcept -> int = 0;
80     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
81         -> const session::internal::Session& = 0;
83     virtual auto Network() const noexcept -> const network::Network& = 0;
84     virtual auto QtRootObject() const noexcept -> QObject* = 0;
86     virtual auto SetMasterKeyTimeout(
87         const std::chrono::seconds& timeout) const noexcept -> void = 0;
88     OPENTXS_NO_EXPORT virtual auto Storage() const noexcept
89         -> const session::Storage& = 0;
91     virtual auto Wallet() const noexcept -> const session::Wallet& = 0;
92
93     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
94         -> session::internal::Session& = 0;
95
96     OPENTXS_NO_EXPORT ~Session() override = default;
97
98 protected:
99     Session() = default;
100
101 private:
102     Session(const Session&) = delete;
103     Session(Session&&) = delete;
104     auto operator=(const Session&) -> Session& = delete;
105     auto operator=(Session&&) -> Session& = delete;
106 };
107 }  // namespace opentxs::api
```

7.28 Storage.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <cstdint>
12 #include <ctime>
13 #include <functional>
14 #include <memory>
15
16 #include "opentxs/blockchain/Types.hpp"
17 #include "opentxs/core/Data.hpp"
18 #include "opentxs/core/Types.hpp"
19 #include "opentxs/core/identifier/Generic.hpp"
20 #include "opentxs/core/identifier/Notary.hpp"
21 #include "opentxs/core/identifier/Nym.hpp"
22 #include "opentxs/core/identifier/UnitDefinition.hpp"
23 #include "opentxs/otx/client/Types.hpp"
24 #include "opentxs/util/Container.hpp"
25 #include "opentxs/util/Numbers.hpp"
26 #include "opentxs/util/Time.hpp"
27 #include "opentxs/util/Types.hpp"
28
29 // NOLINTBEGIN(modernize-concat-nested-namespaces)
30 namespace opentxs  // NOLINT
31 {
32 // inline namespace v1
33 // {
34 namespace api
35 {
36 namespace session
37 {
38 namespace internal
39 {
40 class Storage;
41 }  // namespace internal
42 }  // namespace session
43 }  // namespace api
44
45 namespace identifier
```

```
46 {
47 class Nym;
48 class Notary;
49 class UnitDefinition;
50 }  // namespace identifier
51
52 namespace proto
53 {
54 class Bip47Channel;
55 class Ciphertext;
56 class Contact;
57 class Context;
58 class Credential;
59 class HDAccount;
60 class Issuer;
61 class Nym;
62 class PaymentWorkflow;
63 class PeerReply;
64 class PeerRequest;
65 class Purse;
66 class Seed;
67 class ServerContract;
68 class StorageThread;
69 class UnitDefinition;
70 }  // namespace proto
71
72 using NymLambda = std::function<void(const proto::Nym&)>;
73 using ServerLambda = std::function<void(const proto::ServerContract&)>;
74 using UnitLambda = std::function<void(const proto::UnitDefinition&)>;
75 // }  // namespace v1
76 }  // namespace opentxs
77 // NOLINTEND(modernize-concat-nested-namespaces)
78
79 namespace opentxs::api::session
80 {
81 class Storage
82 {
83 public:
84     using Bip47ChannelList = UnallocatedSet<OTIdentifier>;
85
86     virtual auto AccountAlias(const Identifier& accountID) const
87         -> UnallocatedCString = 0;
88     virtual auto AccountList() const -> ObjectList = 0;
89     virtual auto AccountContract(const Identifier& accountID) const
90         -> OTUnitID = 0;
91     virtual auto AccountIssuer(const Identifier& accountID) const
92         -> OTNymID = 0;
93     virtual auto AccountOwner(const Identifier& accountID) const -> OTNymID = 0;
94     virtual auto AccountServer(const Identifier& accountID) const
95         -> OTNotaryID = 0;
96     virtual auto AccountSigner(const Identifier& accountID) const
97         -> OTNymID = 0;
98     virtual auto AccountUnit(const Identifier& accountID) const -> UnitType = 0;
99     virtual auto AccountsByContract(const identifier::UnitDefinition& contract)
100         const -> UnallocatedSet<OTIdentifier> = 0;
101     virtual auto AccountsByIssuer(const identifier::Nym& issuerNym) const
102         -> UnallocatedSet<OTIdentifier> = 0;
103     virtual auto AccountsByOwner(const identifier::Nym& ownerNym) const
104         -> UnallocatedSet<OTIdentifier> = 0;
105     virtual auto AccountsByServer(const identifier::Notary& server) const
106         -> UnallocatedSet<OTIdentifier> = 0;
107     virtual auto AccountsByUnit(const UnitType unit) const
108         -> UnallocatedSet<OTIdentifier> = 0;
109     virtual auto Bip47Chain(
110         const identifier::Nym& nymID,
111         const Identifier& channelID) const -> UnitType = 0;
112     virtual auto Bip47ChannelsByChain(
113         const identifier::Nym& nymID,
114         const UnitType chain) const -> Bip47ChannelList = 0;
115     virtual auto BlockchainAccountList(
116         const UnallocatedCString& nymID,
117         const UnitType type) const -> UnallocatedSet<UnallocatedCString> = 0;
118     virtual auto BlockchainSubaccountAccountType(
119         const identifier::Nym& owner,
120         const Identifier& id) const -> UnitType = 0;
121     virtual auto BlockchainThreadMap(
122         const identifier::Nym& nym,
123         const Data& txid) const noexcept -> UnallocatedVector<OTIdentifier> = 0;
124     virtual auto BlockchainTransactionList(const identifier::Nym& nym)
125         const noexcept -> UnallocatedVector<OTData> = 0;
126     virtual auto CheckTokenSpent(
127         const identifier::Notary& notary,
128         const identifier::UnitDefinition& unit,
129         const std::uint64_t series,
130         const UnallocatedCString& key) const -> bool = 0;
131     virtual auto ContactAlias(const UnallocatedCString& id) const
132         -> UnallocatedCString = 0;
```

```
133     virtual auto ContactList() const -> ObjectList = 0;
134     virtual auto ContextList(const UnallocatedCString& nymID) const
135         -> ObjectList = 0;
136     virtual auto ContactOwnerNym(const UnallocatedCString& nymID) const
137         -> UnallocatedCString = 0;
138     virtual void ContactSaveIndices() const = 0;
139     virtual auto ContactUpgradeLevel() const -> VersionNumber = 0;
140     virtual auto CreateThread(
141         const UnallocatedCString& nymID,
142         const UnallocatedCString& threadID,
143         const UnallocatedSet<UnallocatedCString>& participants) const
144         -> bool = 0;
145     virtual auto DeleteAccount(const UnallocatedCString& id) const -> bool = 0;
146     virtual auto DefaultNym() const -> OTNymID = 0;
147     virtual auto DefaultSeed() const -> UnallocatedCString = 0;
148     virtual auto DeleteContact(const UnallocatedCString& id) const -> bool = 0;
149     virtual auto DeletePaymentWorkflow(
150         const UnallocatedCString& nymID,
151         const UnallocatedCString& workflowID) const -> bool = 0;
152     virtual auto HashType() const -> std::uint32_t = 0;
153     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
154         -> const internal::Storage& = 0;
155     virtual auto IssuerList(const UnallocatedCString& nymID) const
156         -> ObjectList = 0;
157     virtual auto Load(
158         const UnallocatedCString& accountID,
159         UnallocatedCString& output,
160         UnallocatedCString& alias,
161         const bool checking = false) const -> bool = 0;
162     virtual auto Load(
163         const UnallocatedCString& nymID,
164         const UnallocatedCString& accountID,
165         proto::HDAccount& output,
166         const bool checking = false) const -> bool = 0;
167     virtual auto Load(
168         const identifier::Nym& nymID,
169         const Identifier& channelID,
170         proto::Bip47Channel& output,
171         const bool checking = false) const -> bool = 0;
172     virtual auto Load(
173         const UnallocatedCString& id,
174         proto::Contact& contact,
175         const bool checking = false) const -> bool = 0;
176     virtual auto Load(
177         const UnallocatedCString& id,
178         proto::Contact& contact,
179         UnallocatedCString& alias,
180         const bool checking = false) const -> bool = 0;
181     virtual auto Load(
182         const UnallocatedCString& nym,
183         const UnallocatedCString& id,
184         proto::Context& context,
185         const bool checking = false) const -> bool = 0;
186     virtual auto Load(
187         const UnallocatedCString& id,
188         proto::Credential& cred,
189         const bool checking = false) const -> bool = 0;
190     virtual auto Load(
191         const identifier::Nym& id,
192         proto::Nym& nym,
193         const bool checking = false) const -> bool = 0;
194     virtual auto Load(
195         const identifier::Nym& id,
196         proto::Nym& nym,
197         UnallocatedCString& alias,
198         const bool checking = false) const -> bool = 0;
199     virtual auto LoadNym(
200         const identifier::Nym& id,
201         AllocateOutput destination,
202         const bool checking = false) const -> bool = 0;
203     virtual auto Load(
204         const UnallocatedCString& nymID,
205         const UnallocatedCString& id,
206         proto::Issuer& issuer,
207         const bool checking = false) const -> bool = 0;
208     virtual auto Load(
209         const UnallocatedCString& nymID,
210         const UnallocatedCString& workflowID,
211         proto::PaymentWorkflow& workflow,
212         const bool checking = false) const -> bool = 0;
213     virtual auto Load(
214         const UnallocatedCString& nymID,
215         const UnallocatedCString& id,
216         const otx::client::StorageBox box,
217         UnallocatedCString& output,
218         UnallocatedCString& alias,
219         const bool checking = false) const -> bool = 0;
```

```
220     virtual auto Load(
221         const UnallocatedCString& nymID,
222         const UnallocatedCString& id,
223         const otx::client::StorageBox box,
224         proto::PeerReply& request,
225         const bool checking = false) const -> bool = 0;
226     virtual auto Load(
227         const UnallocatedCString& nymID,
228         const UnallocatedCString& id,
229         const otx::client::StorageBox box,
230         proto::PeerRequest& request,
231         std::time_t& time,
232         const bool checking = false) const -> bool = 0;
233     virtual auto Load(
234         const identifier::Nym& nym,
235         const identifier::Notary& notary,
236         const identifier::UnitDefinition& unit,
237         proto::Purse& output,
238         const bool checking) const -> bool = 0;
239     virtual auto Load(
240         const UnallocatedCString& id,
241         proto::Seed& seed,
242         const bool checking = false) const -> bool = 0;
243     virtual auto Load(
244         const UnallocatedCString& id,
245         proto::Seed& seed,
246         UnallocatedCString& alias,
247         const bool checking = false) const -> bool = 0;
248     virtual auto Load(
249         const identifier::Notary& id,
250         proto::ServerContract& contract,
251         const bool checking = false) const -> bool = 0;
252     virtual auto Load(
253         const identifier::Notary& id,
254         proto::ServerContract& contract,
255         UnallocatedCString& alias,
256         const bool checking = false) const -> bool = 0;
257     virtual auto Load(
258         const UnallocatedCString& nymId,
259         const UnallocatedCString& threadId,
260         proto::StorageThread& thread) const -> bool = 0;
261     virtual auto Load(proto::Ciphertext& output, const bool checking = false)
262         const -> bool = 0;
263     virtual auto Load(
264         const identifier::UnitDefinition& id,
265         proto::UnitDefinition& contract,
266         const bool checking = false) const -> bool = 0;
267     virtual auto Load(
268         const identifier::UnitDefinition& id,
269         proto::UnitDefinition& contract,
270         UnallocatedCString& alias,
271         const bool checking = false) const -> bool = 0;
272     virtual auto LocalNyms() const
273         -> const UnallocatedSet<UnallocatedCString> = 0;
274     virtual void MapPublicNyms(NymLambda& lambda) const = 0;
275     virtual void MapServers(ServerLambda& lambda) const = 0;
276     virtual void MapUnitDefinitions(UnitLambda& lambda) const = 0;
277     virtual auto MarkTokenSpent(
278         const identifier::Notary& notary,
279         const identifier::UnitDefinition& unit,
280         const std::uint64_t series,
281         const UnallocatedCString& key) const -> bool = 0;
282     virtual auto MoveThreadItem(
283         const UnallocatedCString& nymId,
284         const UnallocatedCString& fromThreadID,
285         const UnallocatedCString& toThreadID,
286         const UnallocatedCString& itemID) const -> bool = 0;
287     virtual auto NymBoxList(
288         const UnallocatedCString& nymID,
289         const otx::client::StorageBox box) const -> ObjectList = 0;
290     virtual auto NymList() const -> ObjectList = 0;
291     virtual auto PaymentWorkflowList(const UnallocatedCString& nymID) const
292         -> ObjectList = 0;
293     virtual auto PaymentWorkflowLookup(
294         const UnallocatedCString& nymID,
295         const UnallocatedCString& sourceID) const -> UnallocatedCString = 0;
296     virtual auto PaymentWorkflowsByAccount(
297         const UnallocatedCString& nymID,
298         const UnallocatedCString& accountID) const
299         -> UnallocatedSet<UnallocatedCString> = 0;
300     virtual auto PaymentWorkflowsByState(
301         const UnallocatedCString& nymID,
302         const otx::client::PaymentWorkflowType type,
303         const otx::client::PaymentWorkflowState state) const
304         -> UnallocatedSet<UnallocatedCString> = 0;
305     virtual auto PaymentWorkflowsByUnit(
306         const UnallocatedCString& nymID,
```

```
307         const UnallocatedCString& unitID) const
308         -> UnallocatedSet<UnallocatedCString> = 0;
309     virtual auto PaymentWorkflowState(
310         const UnallocatedCString& nymID,
311         const UnallocatedCString& workflowID) const
312         -> std::pair<
313             otx::client::PaymentWorkflowType,
314             otx::client::PaymentWorkflowState> = 0;
315     virtual auto RelabelThread(
316         const UnallocatedCString& threadID,
317         const UnallocatedCString& label) const -> bool = 0;
318     virtual auto RemoveBlockchainThreadItem(
319         const identifier::Nym& nym,
320         const Identifier& thread,
321         const opentxs::blockchain::Type chain,
322         const Data& txid) const noexcept -> bool = 0;
323     virtual auto RemoveNymBoxItem(
324         const UnallocatedCString& nymID,
325         const otx::client::StorageBox box,
326         const UnallocatedCString& itemID) const -> bool = 0;
327     virtual auto RemoveServer(const UnallocatedCString& id) const -> bool = 0;
328     virtual auto RemoveThreadItem(
329         const identifier::Nym& nym,
330         const Identifier& thread,
331         const UnallocatedCString& id) const -> bool = 0;
332     virtual auto RemoveUnitDefinition(const UnallocatedCString& id) const
333         -> bool = 0;
334     virtual auto RenameThread(
335         const UnallocatedCString& nymId,
336         const UnallocatedCString& threadId,
337         const UnallocatedCString& newID) const -> bool = 0;
338     virtual void RunGC() const = 0;
339     virtual auto ServerAlias(const UnallocatedCString& id) const
340         -> UnallocatedCString = 0;
341     virtual auto ServerList() const -> ObjectList = 0;
342     virtual auto SeedList() const -> ObjectList = 0;
343     virtual auto SetAccountAlias(
344         const UnallocatedCString& id,
345         const UnallocatedCString& alias) const -> bool = 0;
346     virtual auto SetContactAlias(
347         const UnallocatedCString& id,
348         const UnallocatedCString& alias) const -> bool = 0;
349     virtual auto SetDefaultNym(const identifier::Nym& id) const -> bool = 0;
350     virtual auto SetDefaultSeed(const UnallocatedCString& id) const -> bool = 0;
351     virtual auto SetNymAlias(
352         const identifier::Nym& id,
353         const UnallocatedCString& alias) const -> bool = 0;
354     virtual auto SetPeerRequestTime(
355         const UnallocatedCString& nymID,
356         const UnallocatedCString& id,
357         const otx::client::StorageBox box) const -> bool = 0;
358     virtual auto SetReadState(
359         const UnallocatedCString& nymId,
360         const UnallocatedCString& threadId,
361         const UnallocatedCString& itemId,
362         const bool unread) const -> bool = 0;
363     virtual auto SetSeedAlias(
364         const UnallocatedCString& id,
365         const UnallocatedCString& alias) const -> bool = 0;
366     virtual auto SetServerAlias(
367         const identifier::Notary& id,
368         const UnallocatedCString& alias) const -> bool = 0;
369     virtual auto SetThreadAlias(
370         const UnallocatedCString& nymId,
371         const UnallocatedCString& threadId,
372         const UnallocatedCString& alias) const -> bool = 0;
373     virtual auto SetUnitDefinitionAlias(
374         const identifier::UnitDefinition& id,
375         const UnallocatedCString& alias) const -> bool = 0;
376     virtual auto Store(
377         const UnallocatedCString& accountID,
378         const UnallocatedCString& data,
379         const UnallocatedCString& alias,
380         const identifier::Nym& ownerNym,
381         const identifier::Nym& signerNym,
382         const identifier::Nym& issuerNym,
383         const identifier::Notary& server,
384         const identifier::UnitDefinition& contract,
385         const UnitType unit) const -> bool = 0;
386     virtual auto Store(
387         const UnallocatedCString& nymID,
388         const opentxs::identity::wot::claim::ClaimType type,
389         const proto::HDAccount& data) const -> bool = 0;
390     virtual auto Store(
391         const identifier::Nym& nymID,
392         const Identifier& channelID,
393         const proto::Bip47Channel& data) const -> bool = 0;
```

```
394     virtual auto Store(const proto::Contact& data) const -> bool = 0;
395     virtual auto Store(const proto::Context& data) const -> bool = 0;
396     virtual auto Store(const proto::Credential& data) const -> bool = 0;
397     virtual auto Store(
398         const proto::Nym& data,
399         const UnallocatedCString& alias = {}) const -> bool = 0;
400     virtual auto Store(
401         const ReadView& data,
402         const UnallocatedCString& alias = {}) const -> bool = 0;
403     virtual auto Store(
404         const UnallocatedCString& nymID,
405         const proto::Issuer& data) const -> bool = 0;
406     virtual auto Store(
407         const UnallocatedCString& nymID,
408         const proto::PaymentWorkflow& data) const -> bool = 0;
409     virtual auto Store(
410         const UnallocatedCString& nymid,
411         const UnallocatedCString& threadid,
412         const UnallocatedCString& itemid,
413         const std::uint64_t time,
414         const UnallocatedCString& alias,
415         const UnallocatedCString& data,
416         const otx::client::StorageBox box,
417         const UnallocatedCString& account = {}) const -> bool = 0;
418     virtual auto Store(
419         const identifier::Nym& nym,
420         const Identifier& thread,
421         const opentxs::blockchain::Type chain,
422         const Data& txid,
423         const Time time) const noexcept -> bool = 0;
424     virtual auto Store(
425         const proto::PeerReply& data,
426         const UnallocatedCString& nymid,
427         const otx::client::StorageBox box) const -> bool = 0;
428     virtual auto Store(
429         const proto::PeerRequest& data,
430         const UnallocatedCString& nymid,
431         const otx::client::StorageBox box) const -> bool = 0;
432     virtual auto Store(const identifier::Nym& nym, const proto::Purse& purse)
433         const -> bool = 0;
434     virtual auto Store(const proto::Seed& data) const -> bool = 0;
435     virtual auto Store(
436         const proto::ServerContract& data,
437         const UnallocatedCString& alias = {}) const -> bool = 0;
438     virtual auto Store(const proto::Ciphertext& serialized) const -> bool = 0;
439     virtual auto Store(
440         const proto::UnitDefinition& data,
441         const UnallocatedCString& alias = {}) const -> bool = 0;
442     virtual auto ThreadList(
443         const UnallocatedCString& nymID,
444         const bool unreadOnly) const -> ObjectList = 0;
445     virtual auto ThreadAlias(
446         const UnallocatedCString& nymID,
447         const UnallocatedCString& threadID) const -> UnallocatedCString = 0;
448     virtual auto UnaffiliatedBlockchainTransaction(
449         const identifier::Nym& recipient,
450         const Data& txid) const noexcept -> bool = 0;
451     virtual auto UnitDefinitionAlias(const UnallocatedCString& id) const
452         -> UnallocatedCString = 0;
453     virtual auto UnitDefinitionList() const -> ObjectList = 0;
454     virtual auto UnreadCount(
455         const UnallocatedCString& nymId,
456         const UnallocatedCString& threadId) const -> std::size_t = 0;
457     virtual void UpgradeNyms() = 0;
458
459     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
460         -> internal::Storage& = 0;
461
462     OPENTXS_NO_EXPORT virtual ~Storage() = default;
463
464 protected:
465     Storage() = default;
466
467 private:
468     Storage(const Storage&) = delete;
469     Storage(Storage&&) = delete;
470     auto operator=(const Storage&) -> Storage& = delete;
471     auto operator=(Storage&&) -> Storage& = delete;
472 };
473 }  // namespace opentxs::api::session
```

7.29 UI.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
```

```
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 // IWYU pragma: no_include "opentxs/blockchain/BlockchainType.hpp"
9 // IWYU pragma: no_include "opentxs/identity/wot/claim/ClaimType.hpp"
10 // IWYU pragma: no_include "opentxs/core/UnitType.hpp"
11 // IWYU pragma: no_include "opentxs/interface/ui/Blockchains.hpp"
12
13 #include "opentxs/Version.hpp"  // IWYU pragma: associated
14
15 #include <cstddef>
16 #include <iosfwd>
17
18 #include "opentxs/blockchain/Types.hpp"
19 #include "opentxs/core/Types.hpp"
20 #include "opentxs/crypto/Types.hpp"
21 #include "opentxs/interface/ui/Types.hpp"
22 #include "opentxs/util/Types.hpp"
23
24 class QAbstractItemModel;
25
26 // NOLINTBEGIN(modernize-concat-nested-namespaces)
27 namespace opentxs  // NOLINT
28 {
29 // inline namespace v1
30 // {
31 namespace api
32 {
33 namespace session
34 {
35 namespace internal
36 {
37 class UI;
38 }  // namespace internal
39 }  // namespace session
40 }  // namespace api
41
42 namespace identifier
43 {
44 class Nym;
45 class Notary;
46 class UnitDefinition;
47 }  // namespace identifier
48
49 namespace ui
50 {
51 class AccountActivity;
52 class AccountActivityQt;
53 class AccountList;
54 class AccountListQt;
55 class AccountSummary;
56 class AccountSummaryQt;
57 class AccountTree;
58 class AccountTreeQt;
59 class ActivitySummary;
60 class ActivitySummaryQt;
61 class ActivityThread;
62 class ActivityThreadQt;
63 class BlockchainAccountStatus;
64 class BlockchainAccountStatusQt;
65 class BlockchainSelection;
66 class BlockchainSelectionQt;
67 class BlockchainStatistics;
68 class BlockchainStatisticsQt;
69 class Contact;
70 class ContactList;
71 class ContactListQt;
72 class ContactQt;
73 class IdentityManagerQt;
74 class MessagableList;
75 class MessagableListQt;
76 class NymList;
77 class NymListQt;
78 class PayableList;
79 class PayableListQt;
80 class Profile;
81 class ProfileQt;
82 class SeedTree;
83 class SeedTreeQt;
84 class SeedValidator;
85 class UnitList;
86 class UnitListQt;
87 }  // namespace ui
88
```

```
89 class Identifier;
90 // }  // namespace v1
91 }  // namespace opentxs
92 // NOLINTEND(modernize-concat-nested-namespaces)
93
94 namespace opentxs::api::session
95 {
96 class OPENTXS_EXPORT UI
97 {
98 public:
99     virtual auto AccountActivity(
100         const identifier::Nym& nymID,
101         const Identifier& accountID,
102         const SimpleCallback updateCB = {}) const noexcept
103         -> const opentxs::ui::AccountActivity& = 0;
105     virtual auto AccountActivityQt(
106         const identifier::Nym& nymID,
107         const Identifier& accountID,
108         const SimpleCallback updateCB = {}) const noexcept
109         -> opentxs::ui::AccountActivityQt* = 0;
110     virtual auto AccountList(
111         const identifier::Nym& nym,
112         const SimpleCallback updateCB = {}) const noexcept
113         -> const opentxs::ui::AccountList& = 0;
115     virtual auto AccountListQt(
116         const identifier::Nym& nym,
117         const SimpleCallback updateCB = {}) const noexcept
118         -> opentxs::ui::AccountListQt* = 0;
119     virtual auto AccountSummary(
120         const identifier::Nym& nymID,
121         const UnitType currency,
122         const SimpleCallback updateCB = {}) const noexcept
123         -> const opentxs::ui::AccountSummary& = 0;
125     virtual auto AccountSummaryQt(
126         const identifier::Nym& nymID,
127         const UnitType currency,
128         const SimpleCallback updateCB = {}) const noexcept
129         -> opentxs::ui::AccountSummaryQt* = 0;
130     virtual auto AccountTree(
131         const identifier::Nym& nym,
132         const SimpleCallback updateCB = {}) const noexcept
133         -> const opentxs::ui::AccountTree& = 0;
135     virtual auto AccountTreeQt(
136         const identifier::Nym& nym,
137         const SimpleCallback updateCB = {}) const noexcept
138         -> opentxs::ui::AccountTreeQt* = 0;
139     virtual auto ActivitySummary(
140         const identifier::Nym& nymID,
141         const SimpleCallback updateCB = {}) const noexcept
142         -> const opentxs::ui::ActivitySummary& = 0;
144     virtual auto ActivitySummaryQt(
145         const identifier::Nym& nymID,
146         const SimpleCallback updateCB = {}) const noexcept
147         -> opentxs::ui::ActivitySummaryQt* = 0;
148     virtual auto ActivityThread(
149         const identifier::Nym& nymID,
150         const Identifier& threadID,
151         const SimpleCallback updateCB = {}) const noexcept
152         -> const opentxs::ui::ActivityThread& = 0;
154     virtual auto ActivityThreadQt(
155         const identifier::Nym& nymID,
156         const Identifier& threadID,
157         const SimpleCallback updateCB = {}) const noexcept
158         -> opentxs::ui::ActivityThreadQt* = 0;
160     virtual auto BlankModel(const std::size_t columns) const noexcept
161         -> QAbstractItemModel* = 0;
162     virtual auto BlockchainAccountStatus(
163         const identifier::Nym& nymID,
164         const opentxs::blockchain::Type chain,
165         const SimpleCallback updateCB = {}) const noexcept
166         -> const opentxs::ui::BlockchainAccountStatus& = 0;
168     virtual auto BlockchainAccountStatusQt(
169         const identifier::Nym& nymID,
170         const opentxs::blockchain::Type chain,
171         const SimpleCallback updateCB = {}) const noexcept
172         -> opentxs::ui::BlockchainAccountStatusQt* = 0;
173     virtual auto BlockchainIssuerID(const opentxs::blockchain::Type chain)
174         const noexcept -> const identifier::Nym& = 0;
175     virtual auto BlockchainNotaryID(const opentxs::blockchain::Type chain)
176         const noexcept -> const identifier::Notary& = 0;
177     virtual auto BlockchainSelection(
178         const opentxs::ui::Blockchains type,
179         const SimpleCallback updateCB = {}) const noexcept
180         -> const opentxs::ui::BlockchainSelection& = 0;
182     virtual auto BlockchainSelectionQt(
183         const opentxs::ui::Blockchains type,
184         const SimpleCallback updateCB = {}) const noexcept
```

```
185         -> opentxs::ui::BlockchainSelectionQt* = 0;
186     virtual auto BlockchainStatistics(const SimpleCallback updateCB = {})
187         const noexcept -> const opentxs::ui::BlockchainStatistics& = 0;
188     virtual auto BlockchainStatisticsQt(const SimpleCallback updateCB = {})
189         const noexcept -> opentxs::ui::BlockchainStatisticsQt* = 0;
190     virtual auto BlockchainUnitID(const opentxs::blockchain::Type chain)
191         const noexcept -> const identifier::UnitDefinition& = 0;
192     virtual auto Contact(
193         const Identifier& contactID,
194         const SimpleCallback updateCB = {}) const noexcept
195         -> const opentxs::ui::Contact& = 0;
196     virtual auto ContactQt(
197         const Identifier& contactID,
198         const SimpleCallback updateCB = {}) const noexcept
199         -> opentxs::ui::ContactQt* = 0;
200     virtual auto ContactList(
201         const identifier::Nym& nymID,
202         const SimpleCallback updateCB = {}) const noexcept
203         -> const opentxs::ui::ContactList& = 0;
205     virtual auto ContactListQt(
206         const identifier::Nym& nymID,
207         const SimpleCallback updateCB = {}) const noexcept
208         -> opentxs::ui::ContactListQt* = 0;
210     virtual auto IdentityManagerQt() const noexcept
211         -> opentxs::ui::IdentityManagerQt* = 0;
212     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
213         -> const internal::UI& = 0;
214     virtual auto MessagableList(
215         const identifier::Nym& nymID,
216         const SimpleCallback updateCB = {}) const noexcept
217         -> const opentxs::ui::MessagableList& = 0;
219     virtual auto MessagableListQt(
220         const identifier::Nym& nymID,
221         const SimpleCallback updateCB = {}) const noexcept
222         -> opentxs::ui::MessagableListQt* = 0;
223     virtual auto NymList(const SimpleCallback updateCB = {}) const noexcept
224         -> const opentxs::ui::NymList& = 0;
226     virtual auto NymListQt(const SimpleCallback updateCB = {}) const noexcept
227         -> opentxs::ui::NymListQt* = 0;
228     virtual auto PayableList(
229         const identifier::Nym& nymID,
230         const UnitType currency,
231         const SimpleCallback updateCB = {}) const noexcept
232         -> const opentxs::ui::PayableList& = 0;
234     virtual auto PayableListQt(
235         const identifier::Nym& nymID,
236         const UnitType currency,
237         const SimpleCallback updateCB = {}) const noexcept
238         -> opentxs::ui::PayableListQt* = 0;
239     virtual auto Profile(
240         const identifier::Nym& nymID,
241         const SimpleCallback updateCB = {}) const noexcept
242         -> const opentxs::ui::Profile& = 0;
244     virtual auto ProfileQt(
245         const identifier::Nym& nymID,
246         const SimpleCallback updateCB = {}) const noexcept
247         -> opentxs::ui::ProfileQt* = 0;
248     virtual auto SeedTree(const SimpleCallback updateCB = {}) const noexcept
249         -> const opentxs::ui::SeedTree& = 0;
251     virtual auto SeedTreeQt(const SimpleCallback updateCB = {}) const noexcept
252         -> opentxs::ui::SeedTreeQt* = 0;
254     virtual auto SeedValidator(
255         const opentxs::crypto::SeedStyle type,
256         const opentxs::crypto::Language lang) const noexcept
257         -> const opentxs::ui::SeedValidator* = 0;
258     virtual auto UnitList(
259         const identifier::Nym& nym,
260         const SimpleCallback updateCB = {}) const noexcept
261         -> const opentxs::ui::UnitList& = 0;
263     virtual auto UnitListQt(
264         const identifier::Nym& nym,
265         const SimpleCallback updateCB = {}) const noexcept
266         -> opentxs::ui::UnitListQt* = 0;
267
268     OPENTXS_NO_EXPORT virtual auto Internal() noexcept -> internal::UI& = 0;
269
270     OPENTXS_NO_EXPORT virtual ~UI() = default;
271
272 protected:
273     UI() = default;
274
275 private:
276     UI(const UI&) = delete;
277     UI(UI&&) = delete;
278     auto operator=(const UI&) -> UI& = delete;
279     auto operator=(UI&&) -> UI& = delete;
280 };
```

```
281 }  // namespace opentxs::api::session
```

7.30 Wallet.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/identity/IdentityType.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include <chrono>
13 #include <cstdint>
14 #include <ctime>
15 #include <memory>
16 #include <tuple>
17
18 #include "opentxs/core/contract/BasketContract.hpp"
19 #include "opentxs/core/contract/ServerContract.hpp"
20 #include "opentxs/core/contract/Unit.hpp"
21 #include "opentxs/identity/Nym.hpp"
22 #include "opentxs/identity/Types.hpp"
23 #include "opentxs/identity/wot/claim/Types.hpp"
24 #include "opentxs/otx/client/Types.hpp"
25 #include "opentxs/util/Container.hpp"
26 #include "opentxs/util/Types.hpp"
27
28 // NOLINTBEGIN(modernize-concat-nested-namespaces)
29 namespace opentxs  // NOLINT
30 {
31 // inline namespace v1
32 // {
33 namespace api
34 {
35 namespace crypto
36 {
37 class Parameters;
38 }  // namespace crypto
39
40 namespace session
41 {
42 namespace internal
43 {
44 class Wallet;
45 }  // namespace internal
46 }  // namespace session
47 }  // namespace api
48
49 namespace display
50 {
51 class Definition;
52 }  // namespace display
53
54 namespace otx
55 {
56 namespace blind
57 {
58 class Purse;
59 }  // namespace blind
60
61 namespace context
62 {
63 class Base;
64 class Client;
65 class Server;
66 }  // namespace context
67 }  // namespace otx
68
69 class Account;
70 class NymData;
71 class PeerObject;
72 // }  // namespace v1
73 }  // namespace opentxs
74 // NOLINTEND(modernize-concat-nested-namespaces)
75
76 namespace opentxs
77 {
79 using AccountInfo = std::tuple<OTIdentifier, OTNymID, OTNotaryID, OTUnitID>;
80 }  // namespace opentxs
```

```
81
82 namespace opentxs::api::session
83 {
94 class OPENTXS_EXPORT Wallet
95 {
96 public:
97     using AccountCallback = std::function<void(const Account&)>;
98
99     virtual auto AccountPartialMatch(const UnallocatedCString& hint) const
100         -> OTIdentifier = 0;
101     virtual auto DeleteAccount(const Identifier& accountID) const -> bool = 0;
102     virtual auto UpdateAccount(
103         const Identifier& accountID,
104         const otx::context::Server&,
105         const String& serialized,
106         const PasswordPrompt& reason) const -> bool = 0;
107     virtual auto UpdateAccount(
108         const Identifier& accountID,
109         const otx::context::Server&,
110         const String& serialized,
111         const UnallocatedCString& label,
112         const PasswordPrompt& reason) const -> bool = 0;
113     [[deprecated]] virtual auto ImportAccount(
114         std::unique_ptr<opentxs::Account>& imported) const -> bool = 0;
115
127     virtual auto Context(
128         const identifier::Notary& notaryID,
129         const identifier::Nym& clientNymID) const
130         -> std::shared_ptr<const otx::context::Base> = 0;
131
139     virtual auto ClientContext(const identifier::Nym& remoteNymID) const
140         -> std::shared_ptr<const otx::context::Client> = 0;
141
142     virtual auto DefaultNym() const noexcept
143         -> std::pair<OTNymID, std::size_t> = 0;
144
153     virtual auto ServerContext(
154         const identifier::Nym& localNymID,
155         const Identifier& remoteID) const
156         -> std::shared_ptr<const otx::context::Server> = 0;
157
159     virtual auto IssuerList(const identifier::Nym& nymID) const
160         -> UnallocatedSet<OTNymID> = 0;
161
162     virtual auto IsLocalNym(const UnallocatedCString& id) const -> bool = 0;
163     virtual auto IsLocalNym(const identifier::Nym& id) const -> bool = 0;
164
165     virtual auto LocalNymCount() const -> std::size_t = 0;
166
167     virtual auto LocalNyms() const -> UnallocatedSet<OTNymID> = 0;
168
188     virtual auto Nym(
189         const identifier::Nym& id,
190         const std::chrono::milliseconds& timeout = 0ms) const -> Nym_p = 0;
191
199     virtual auto Nym(const ReadView& bytes) const -> Nym_p = 0;
200
201     virtual auto Nym(
202         const identity::Type type,
203         const PasswordPrompt& reason,
204         const UnallocatedCString& name = {}) const -> Nym_p = 0;
205     virtual auto Nym(
206         const opentxs::crypto::Parameters& parameters,
207         const PasswordPrompt& reason,
208         const UnallocatedCString& name = {}) const -> Nym_p = 0;
209     virtual auto Nym(
210         const PasswordPrompt& reason,
211         const UnallocatedCString& name = {}) const -> Nym_p = 0;
212     virtual auto Nym(
213         const opentxs::crypto::Parameters& parameters,
214         const identity::Type type,
215         const PasswordPrompt& reason,
216         const UnallocatedCString& name = {}) const -> Nym_p = 0;
217
218     virtual auto mutable_Nym(
219         const identifier::Nym& id,
220         const PasswordPrompt& reason) const -> NymData = 0;
221
222     virtual auto Nymfile(
223         const identifier::Nym& id,
224         const PasswordPrompt& reason) const
225         -> std::unique_ptr<const opentxs::NymFile> = 0;
226
227     virtual auto NymByIDPartialMatch(const UnallocatedCString& partialId) const
228         -> Nym_p = 0;
229
232     virtual auto NymList() const -> ObjectList = 0;
```

```
233
234     virtual auto NymNameByIndex(const std::size_t index, String& name) const
235         -> bool = 0;
236
245     virtual auto PeerReply(
246         const identifier::Nym& nym,
247         const Identifier& reply,
248         const otx::client::StorageBox& box,
249         AllocateOutput destination) const -> bool = 0;
250
261     virtual auto PeerReplyComplete(
262         const identifier::Nym& nym,
263         const Identifier& replyOrRequest) const -> bool = 0;
264
274     virtual auto PeerReplyCreateRollback(
275         const identifier::Nym& nym,
276         const Identifier& request,
277         const Identifier& reply) const -> bool = 0;
278
283     virtual auto PeerReplySent(const identifier::Nym& nym) const
284         -> ObjectList = 0;
285
290     virtual auto PeerReplyIncoming(const identifier::Nym& nym) const
291         -> ObjectList = 0;
292
297     virtual auto PeerReplyFinished(const identifier::Nym& nym) const
298         -> ObjectList = 0;
299
304     virtual auto PeerReplyProcessed(const identifier::Nym& nym) const
305         -> ObjectList = 0;
306
320     virtual auto PeerReplyReceive(
321         const identifier::Nym& nym,
322         const PeerObject& reply) const -> bool = 0;
323
332     virtual auto PeerRequest(
333         const identifier::Nym& nym,
334         const Identifier& request,
335         const otx::client::StorageBox& box,
336         std::time_t& time,
337         AllocateOutput destination) const -> bool = 0;
338
348     virtual auto PeerRequestComplete(
349         const identifier::Nym& nym,
350         const Identifier& reply) const -> bool = 0;
351
360     virtual auto PeerRequestCreateRollback(
361         const identifier::Nym& nym,
362         const Identifier& request) const -> bool = 0;
363
370     virtual auto PeerRequestDelete(
371         const identifier::Nym& nym,
372         const Identifier& request,
373         const otx::client::StorageBox& box) const -> bool = 0;
374
379     virtual auto PeerRequestSent(const identifier::Nym& nym) const
380         -> ObjectList = 0;
381
386     virtual auto PeerRequestIncoming(const identifier::Nym& nym) const
387         -> ObjectList = 0;
388
393     virtual auto PeerRequestFinished(const identifier::Nym& nym) const
394         -> ObjectList = 0;
395
400     virtual auto PeerRequestProcessed(const identifier::Nym& nym) const
401         -> ObjectList = 0;
402
412     virtual auto PeerRequestReceive(
413         const identifier::Nym& nym,
414         const PeerObject& request) const -> bool = 0;
415
422     virtual auto PeerRequestUpdate(
423         const identifier::Nym& nym,
424         const Identifier& request,
425         const otx::client::StorageBox& box) const -> bool = 0;
426
427     virtual auto Purse(
428         const identifier::Nym& nym,
429         const identifier::Notary& server,
430         const identifier::UnitDefinition& unit,
431         const bool checking = false) const -> const otx::blind::Purse& = 0;
432
441     virtual auto RemoveServer(const identifier::Notary& id) const -> bool = 0;
442
451     virtual auto RemoveUnitDefinition(
452         const identifier::UnitDefinition& id) const -> bool = 0;
453
```

```
472     virtual auto Server(
473         const identifier::Notary& id,
474         const std::chrono::milliseconds& timeout = std::chrono::milliseconds(
475             0)) const noexcept(false) -> OTServerContract = 0;
476
482     virtual auto Server(const ReadView& contract) const noexcept(false)
483         -> OTServerContract = 0;
484
494     virtual auto Server(
495         const UnallocatedCString& nymid,
496         const UnallocatedCString& name,
497         const UnallocatedCString& terms,
498         const UnallocatedList<contract::Server::Endpoint>& endpoints,
499         const PasswordPrompt& reason,
500         const VersionNumber version) const noexcept(false)
501         -> OTServerContract = 0;
502
505     virtual auto ServerList() const -> ObjectList = 0;
506
507     virtual auto SetDefaultNym(const identifier::Nym& id) const noexcept
508         -> bool = 0;
518     virtual auto SetNymAlias(
519         const identifier::Nym& id,
520         const UnallocatedCString& alias) const -> bool = 0;
521
531     virtual auto SetServerAlias(
532         const identifier::Notary& id,
533         const UnallocatedCString& alias) const -> bool = 0;
534
544     virtual auto SetUnitDefinitionAlias(
545         const identifier::UnitDefinition& id,
546         const UnallocatedCString& alias) const -> bool = 0;
547
551     virtual auto UnitDefinitionList() const -> ObjectList = 0;
552
571     virtual auto UnitDefinition(
572         const identifier::UnitDefinition& id,
573         const std::chrono::milliseconds& timeout = std::chrono::milliseconds(
574             0)) const noexcept(false) -> OTUnitDefinition = 0;
580     virtual auto UnitDefinition(const ReadView contract) const noexcept(false)
581         -> OTUnitDefinition = 0;
582     virtual auto BasketContract(
583         const identifier::UnitDefinition& id,
584         const std::chrono::milliseconds& timeout = std::chrono::milliseconds(
585             0)) const noexcept(false) -> OTBasketContract = 0;
586
595     virtual auto CurrencyContract(
596         const UnallocatedCString& nymid,
597         const UnallocatedCString& shortname,
598         const UnallocatedCString& terms,
599         const UnitType unitOfAccount,
600         const Amount& redemptionIncrement,
601         const PasswordPrompt& reason) const noexcept(false)
602         -> OTUnitDefinition = 0;
603     virtual auto CurrencyContract(
604         const UnallocatedCString& nymid,
605         const UnallocatedCString& shortname,
606         const UnallocatedCString& terms,
607         const UnitType unitOfAccount,
608         const Amount& redemptionIncrement,
609         const display::Definition& displayDefinition,
610         const PasswordPrompt& reason) const noexcept(false)
611         -> OTUnitDefinition = 0;
612     virtual auto CurrencyContract(
613         const UnallocatedCString& nymid,
614         const UnallocatedCString& shortname,
615         const UnallocatedCString& terms,
616         const UnitType unitOfAccount,
617         const Amount& redemptionIncrement,
618         const VersionNumber version,
619         const PasswordPrompt& reason) const noexcept(false)
620         -> OTUnitDefinition = 0;
621     virtual auto CurrencyContract(
622         const UnallocatedCString& nymid,
623         const UnallocatedCString& shortname,
624         const UnallocatedCString& terms,
625         const UnitType unitOfAccount,
626         const Amount& redemptionIncrement,
627         const display::Definition& displayDefinition,
628         const VersionNumber version,
629         const PasswordPrompt& reason) const noexcept(false)
630         -> OTUnitDefinition = 0;
631
640     virtual auto SecurityContract(
641         const UnallocatedCString& nymid,
642         const UnallocatedCString& shortname,
643         const UnallocatedCString& terms,
```

```
644         const UnitType unitOfAccount,
645         const PasswordPrompt& reason,
646         const display::Definition& displayDefinition,
647         const Amount& redemptionIncrement,
648         const VersionNumber version = contract::Unit::DefaultVersion) const
649         noexcept(false) -> OTUnitDefinition = 0;
650
651     virtual auto CurrencyTypeBasedOnUnitType(
652         const identifier::UnitDefinition& contractID) const -> UnitType = 0;
653
654     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
655         -> const session::internal::Wallet& = 0;
656     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
657         -> session::internal::Wallet& = 0;
658
659     OPENTXS_NO_EXPORT virtual ~Wallet() = default;
660
661 protected:
662     Wallet() = default;
663
664 private:
665     Wallet(const Wallet&) = delete;
666     Wallet(Wallet&&) = delete;
667     auto operator=(const Wallet&) -> Wallet& = delete;
668     auto operator=(Wallet&&) -> Wallet& = delete;
669 };
670 }  // namespace opentxs::api::session
```

7.31 Workflow.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/otx/blind/Purse.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include <memory>
13
14 #include "opentxs/core/identifier/Generic.hpp"
15 #include "opentxs/otx/client/Types.hpp"
16 #include "opentxs/util/Container.hpp"
17 #include "opentxs/util/Numbers.hpp"
18
19 // NOLINTBEGIN(modernize-concat-nested-namespaces)
20 namespace opentxs  // NOLINT
21 {
22 // inline namespace v1
23 // {
24 namespace api
25 {
26 namespace session
27 {
28 namespace internal
29 {
30 class Workflow;
31 }  // namespace internal
32 }  // namespace session
33
34 class Session;
35 }  // namespace api
36
37 namespace identifier
38 {
39 class Nym;
40 class Notary;
41 class UnitDefinition;
42 }  // namespace identifier
43
44 namespace otx
45 {
46 namespace blind
47 {
48 class Purse;
49 }  // namespace blind
50 }  // namespace otx
51
52 namespace proto
53 {
```

```
54 class PaymentWorkflow;
55 class Purse;
56 }  // namespace proto
57
58 class OTTransaction;
59 // }  // namespace v1
60 }  // namespace opentxs
61 // NOLINTEND(modernize-concat-nested-namespaces)
62
63 namespace opentxs::api::session
64 {
143 class OPENTXS_EXPORT Workflow
144 {
145 public:
146     using Cheque = std::pair<
147         otx::client::PaymentWorkflowState,
148         std::unique_ptr<opentxs::Cheque>>;
149     using Purse =
150         std::pair<otx::client::PaymentWorkflowState, otx::blind::Purse>;
151     using Transfer = std::
152         pair<otx::client::PaymentWorkflowState, std::unique_ptr<opentxs::Item>>;
153
154     OPENTXS_NO_EXPORT static auto ContainsCash(
155         const proto::PaymentWorkflow& workflow) -> bool;
156     OPENTXS_NO_EXPORT static auto ContainsCheque(
157         const proto::PaymentWorkflow& workflow) -> bool;
158     OPENTXS_NO_EXPORT static auto ContainsTransfer(
159         const proto::PaymentWorkflow& workflow) -> bool;
160     OPENTXS_NO_EXPORT static auto ExtractCheque(
161         const proto::PaymentWorkflow& workflow) -> UnallocatedCString;
162     OPENTXS_NO_EXPORT static auto ExtractPurse(
163         const proto::PaymentWorkflow& workflow,
164         proto::Purse& out) -> bool;
165     OPENTXS_NO_EXPORT static auto ExtractTransfer(
166         const proto::PaymentWorkflow& workflow) -> UnallocatedCString;
167     OPENTXS_NO_EXPORT static auto InstantiateCheque(
168         const api::Session& api,
169         const proto::PaymentWorkflow& workflow) -> Cheque;
170     OPENTXS_NO_EXPORT static auto InstantiatePurse(
171         const api::Session& api,
172         const proto::PaymentWorkflow& workflow) -> Purse;
173     OPENTXS_NO_EXPORT static auto InstantiateTransfer(
174         const api::Session& api,
175         const proto::PaymentWorkflow& workflow) -> Transfer;
176     OPENTXS_NO_EXPORT static auto UUID(
177         const api::Session& api,
178         const proto::PaymentWorkflow& workflown) -> OTIdentifier;
179     static auto UUID(
180         const api::Session& api,
181         const Identifier& notary,
182         const TransactionNumber& number) -> OTIdentifier;
183
185     virtual auto AbortTransfer(
186         const identifier::Nym& nymID,
187         const Item& transfer,
188         const Message& reply) const -> bool = 0;
190     virtual auto AcceptTransfer(
191         const identifier::Nym& nymID,
192         const identifier::Notary& notaryID,
193         const OTTransaction& pending,
194         const Message& reply) const -> bool = 0;
196     virtual auto AcknowledgeTransfer(
197         const identifier::Nym& nymID,
198         const Item& transfer,
199         const Message& reply) const -> bool = 0;
200     virtual auto AllocateCash(
201         const identifier::Nym& id,
202         const otx::blind::Purse& purse) const -> OTIdentifier = 0;
204     virtual auto CancelCheque(
205         const opentxs::Cheque& cheque,
206         const Message& request,
207         const Message* reply) const -> bool = 0;
209     virtual auto ClearCheque(
210         const identifier::Nym& recipientNymID,
211         const OTTransaction& receipt) const -> bool = 0;
213     virtual auto ClearTransfer(
214         const identifier::Nym& nymID,
215         const identifier::Notary& notaryID,
216         const OTTransaction& receipt) const -> bool = 0;
218     virtual auto CompleteTransfer(
219         const identifier::Nym& nymID,
220         const identifier::Notary& notaryID,
221         const OTTransaction& receipt,
222         const Message& reply) const -> bool = 0;
225     virtual auto ConveyTransfer(
226         const identifier::Nym& nymID,
227         const identifier::Notary& notaryID,
```

```
228         const OTTransaction& pending) const -> OTIdentifier = 0;
231     virtual auto CreateTransfer(const Item& transfer, const Message& request)
232         const -> OTIdentifier = 0;
234     virtual auto DepositCheque(
235         const identifier::Nym& nymID,
236         const Identifier& accountID,
237         const opentxs::Cheque& cheque,
238         const Message& request,
239         const Message* reply) const -> bool = 0;
241     virtual auto ExpireCheque(
242         const identifier::Nym& nymID,
243         const opentxs::Cheque& cheque) const -> bool = 0;
245     virtual auto ExportCheque(const opentxs::Cheque& cheque) const -> bool = 0;
247     virtual auto FinishCheque(
248         const opentxs::Cheque& cheque,
249         const Message& request,
250         const Message* reply) const -> bool = 0;
252     virtual auto ImportCheque(
253         const identifier::Nym& nymID,
254         const opentxs::Cheque& cheque) const -> OTIdentifier = 0;
255     virtual auto InstantiateCheque(
256         const identifier::Nym& nymID,
257         const Identifier& workflowID) const -> Cheque = 0;
258     virtual auto InstantiatePurse(
259         const identifier::Nym& nymID,
260         const Identifier& workflowID) const -> Purse = 0;
261     OPENTXS_NO_EXPORT virtual auto Internal() const noexcept
262         -> const internal::Workflow& = 0;
263     virtual auto List(
264         const identifier::Nym& nymID,
265         const otx::client::PaymentWorkflowType type,
266         const otx::client::PaymentWorkflowState state) const
267         -> UnallocatedSet<OTIdentifier> = 0;
268     virtual auto LoadCheque(
269         const identifier::Nym& nymID,
270         const Identifier& chequeID) const -> Cheque = 0;
271     virtual auto LoadChequeByWorkflow(
272         const identifier::Nym& nymID,
273         const Identifier& workflowID) const -> Cheque = 0;
274     virtual auto LoadTransfer(
275         const identifier::Nym& nymID,
276         const Identifier& transferID) const -> Transfer = 0;
277     virtual auto LoadTransferByWorkflow(
278         const identifier::Nym& nymID,
279         const Identifier& workflowID) const -> Transfer = 0;
281     OPENTXS_NO_EXPORT virtual auto LoadWorkflow(
282         const identifier::Nym& nymID,
283         const Identifier& workflowID,
284         proto::PaymentWorkflow& out) const -> bool = 0;
285     virtual auto ReceiveCash(
286         const identifier::Nym& receiver,
287         const otx::blind::Purse& purse,
288         const Message& message) const -> OTIdentifier = 0;
290     virtual auto ReceiveCheque(
291         const identifier::Nym& nymID,
292         const opentxs::Cheque& cheque,
293         const Message& message) const -> OTIdentifier = 0;
294     virtual auto SendCash(
295         const identifier::Nym& sender,
296         const identifier::Nym& recipient,
297         const Identifier& workflowID,
298         const Message& request,
299         const Message* reply) const -> bool = 0;
301     virtual auto SendCheque(
302         const opentxs::Cheque& cheque,
303         const Message& request,
304         const Message* reply) const -> bool = 0;
305     virtual auto WorkflowParty(
306         const identifier::Nym& nymID,
307         const Identifier& workflowID,
308         const int index) const -> const UnallocatedCString = 0;
309     virtual auto WorkflowPartySize(
310         const identifier::Nym& nymID,
311         const Identifier& workflowID,
312         int& partysize) const -> bool = 0;
313     virtual auto WorkflowState(
314         const identifier::Nym& nymID,
315         const Identifier& workflowID) const
316         -> otx::client::PaymentWorkflowState = 0;
317     virtual auto WorkflowType(
318         const identifier::Nym& nymID,
319         const Identifier& workflowID) const
320         -> otx::client::PaymentWorkflowType = 0;
322     virtual auto WorkflowsByAccount(
323         const identifier::Nym& nymID,
324         const Identifier& accountID) const
325         -> UnallocatedVector<OTIdentifier> = 0;
```

```
327     virtual auto WriteCheque(const opentxs::Cheque& cheque) const
328         -> OTIdentifier = 0;
329
330     OPENTXS_NO_EXPORT virtual auto Internal() noexcept
331         -> internal::Workflow& = 0;
332
333     OPENTXS_NO_EXPORT virtual ~Workflow() = default;
334
335 protected:
336     Workflow() = default;
337
338 private:
339     Workflow(const Workflow&) = delete;
340     Workflow(Workflow&&) = delete;
341     auto operator=(const Workflow&) -> Workflow& = delete;
342     auto operator=(Workflow&&) -> Workflow& = delete;
343 };
344 }  // namespace opentxs::api::session
```

7.32 Settings.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstdint>
11
12 #include "opentxs/util/Container.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 class Flag;
20 class String;
21 // }  // namespace v1
22 }  // namespace opentxs
23 // NOLINTEND(modernize-concat-nested-namespaces)
24
25 namespace opentxs::api
26 {
31 class OPENTXS_EXPORT Settings
32 {
33 public:
34     virtual void SetConfigFilePath(const String& strConfigFilePath) const = 0;
35     virtual auto HasConfigFilePath() const -> bool = 0;
36
45     virtual auto Load() const -> bool = 0;
46     virtual auto Save() const -> bool = 0;
48
50     virtual auto IsLoaded() const -> const Flag& = 0;
51
52     // Configuration Helpers
53     //
54
56     virtual auto IsEmpty() const -> bool = 0;
57
66     virtual auto Check_str(
67         const String& strSection,
68         const String& strKey,
69         String& out_strResult,
70         bool& out_bKeyExist) const -> bool = 0;
71     virtual auto Check_long(
72         const String& strSection,
73         const String& strKey,
74         std::int64_t& out_lResult,
75         bool& out_bKeyExist) const -> bool = 0;
76     virtual auto Check_bool(
77         const String& strSection,
78         const String& strKey,
79         bool& out_bResult,
80         bool& out_bKeyExist) const -> bool = 0;
82
91     virtual auto Set_str(
92         const String& strSection,
93         const String& strKey,
94         const String& strValue,
```

```
95          bool& out_bNewOrUpdate) const -> bool = 0;
96      virtual auto Set_str(
97          const String& strSection,
98          const String& strKey,
99          const String& strValue,
100          bool& out_bNewOrUpdate,
101          const String& strComment) const -> bool = 0;
102      virtual auto Set_long(
103          const String& strSection,
104          const String& strKey,
105          const std::int64_t& lValue,
106          bool& out_bNewOrUpdate) const -> bool = 0;
107      virtual auto Set_long(
108          const String& strSection,
109          const String& strKey,
110          const std::int64_t& lValue,
111          bool& out_bNewOrUpdate,
112          const String& strComment) const -> bool = 0;
113      virtual auto Set_bool(
114          const String& strSection,
115          const String& strKey,
116          const bool& bValue,
117          bool& out_bNewOrUpdate) const -> bool = 0;
118      virtual auto Set_bool(
119          const String& strSection,
120          const String& strKey,
121          const bool& bValue,
122          bool& out_bNewOrUpdate,
123          const String& strComment) const -> bool = 0;
125
128      virtual auto CheckSetSection(
129          const String& strSection,
130          const String& strComment,
131          bool& out_bIsNewSection) const -> bool = 0;
132
136      virtual auto CheckSet_str(
137          const String& strSection,
138          const String& strKey,
139          const String& strDefault,
140          UnallocatedCString& out_strResult,
141          bool& out_bIsNew) const -> bool = 0;
142      virtual auto CheckSet_str(
143          const String& strSection,
144          const String& strKey,
145          const String& strDefault,
146          UnallocatedCString& out_strResult,
147          bool& out_bIsNew,
148          const String& strComment) const -> bool = 0;
149      virtual auto CheckSet_str(
150          const String& strSection,
151          const String& strKey,
152          const String& strDefault,
153          String& out_strResult,
154          bool& out_bIsNew) const -> bool = 0;
155      virtual auto CheckSet_str(
156          const String& strSection,
157          const String& strKey,
158          const String& strDefault,
159          String& out_strResult,
160          bool& out_bIsNew,
161          const String& strComment) const -> bool = 0;
162      virtual auto CheckSet_long(
163          const String& strSection,
164          const String& strKey,
165          const std::int64_t& lDefault,
166          std::int64_t& out_lResult,
167          bool& out_bIsNew) const -> bool = 0;
168      virtual auto CheckSet_long(
169          const String& strSection,
170          const String& strKey,
171          const std::int64_t& lDefault,
172          std::int64_t& out_lResult,
173          bool& out_bIsNew,
174          const String& strComment) const -> bool = 0;
175      virtual auto CheckSet_bool(
176          const String& strSection,
177          const String& strKey,
178          const bool& bDefault,
179          bool& out_bResult,
180          bool& out_bIsNew) const -> bool = 0;
181      virtual auto CheckSet_bool(
182          const String& strSection,
183          const String& strKey,
184          const bool& bDefault,
185          bool& out_bResult,
186          bool& out_bIsNew,
187          const String& strComment) const -> bool = 0;
```

```
189
191     virtual auto SetOption_bool(
192         const String& strSection,
193         const String& strKey,
194         bool& bVariableName) const -> bool = 0;
195
196     virtual auto Reset() -> bool = 0;
197
198     OPENTXS_NO_EXPORT virtual ~Settings() = default;
199
200 protected:
201     Settings() = default;
202
203 private:
204     Settings(const Settings&) = delete;
205     Settings(Settings&&) = delete;
206     auto operator=(const Settings&) -> Settings& = delete;
207     auto operator=(Settings&&) -> Settings& = delete;
208 };
209 }  // namespace opentxs::api
```

7.33 AccountActivity.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <tuple>
11
12 #include "opentxs/blockchain/Types.hpp"
13 #include "opentxs/core/Types.hpp"
14 #include "opentxs/interface/ui/List.hpp"
15 #include "opentxs/util/SharedPimpl.hpp"
16
17 // NOLINTBEGIN(modernize-concat-nested-namespaces)
18 namespace opentxs  // NOLINT
19 {
20 // inline namespace v1
21 // {
22 namespace ui
23 {
24 class AccountActivity;
25 class BalanceItem;
26 }  // namespace ui
27
28 class Amount;
29 // }  // namespace v1
30 }  // namespace opentxs
31 // NOLINTEND(modernize-concat-nested-namespaces)
32
33 namespace opentxs::ui
34 {
39 class OPENTXS_EXPORT AccountActivity : virtual public List
40 {
41 public:
42     using Scale = unsigned int;
43
45     virtual auto AccountID() const noexcept -> UnallocatedCString = 0;
47     virtual auto Balance() const noexcept -> const Amount = 0;
49     virtual auto BalancePolarity() const noexcept -> int = 0;
51     virtual auto ContractID() const noexcept -> UnallocatedCString = 0;
53
57     virtual auto DepositAddress() const noexcept -> UnallocatedCString = 0;
58     virtual auto DepositAddress(const blockchain::Type chain) const noexcept
59         -> UnallocatedCString = 0;
61     virtual auto DepositChains() const noexcept
62         -> UnallocatedVector<blockchain::Type> = 0;
64     virtual auto DisplayBalance() const noexcept -> UnallocatedCString = 0;
66     virtual auto DisplayUnit() const noexcept -> UnallocatedCString = 0;
67     virtual auto First() const noexcept
68         -> opentxs::SharedPimpl<opentxs::ui::BalanceItem> = 0;
70     virtual auto Name() const noexcept -> UnallocatedCString = 0;
71     virtual auto Next() const noexcept
72         -> opentxs::SharedPimpl<opentxs::ui::BalanceItem> = 0;
74     virtual auto NotaryID() const noexcept -> UnallocatedCString = 0;
76     virtual auto NotaryName() const noexcept -> UnallocatedCString = 0;
78
87     virtual auto Send(
```

```
88          const Identifier& contact,
89          const Amount& amount,
90          const UnallocatedCString& memo = {}) const noexcept -> bool = 0;
91      virtual auto Send(
92          const Identifier& contact,
93          const UnallocatedCString& amount,
94          const UnallocatedCString& memo = {},
95          Scale scale = 0) const noexcept -> bool = 0;
96      virtual auto Send(
97          const UnallocatedCString& address,
98          const Amount& amount,
99          const UnallocatedCString& memo = {}) const noexcept -> bool = 0;
100     virtual auto Send(
101         const UnallocatedCString& address,
102         const UnallocatedCString& amount,
103         const UnallocatedCString& memo = {},
104         Scale scale = 0) const noexcept -> bool = 0;
106
108     virtual auto SyncPercentage() const noexcept -> double = 0;
110     virtual auto SyncProgress() const noexcept -> std::pair<int, int> = 0;
112     virtual auto Type() const noexcept -> AccountType = 0;
114     virtual auto Unit() const noexcept -> UnitType = 0;
116     virtual auto ValidateAddress(const UnallocatedCString& text) const noexcept
            -> bool = 0;
119     virtual auto ValidateAmount(const UnallocatedCString& text) const noexcept
120         -> UnallocatedCString = 0;
121
122     ~AccountActivity() override = default;
123
124 protected:
125     AccountActivity() noexcept = default;
126
127 private:
128     AccountActivity(const AccountActivity&) = delete;
129     AccountActivity(AccountActivity&&) = delete;
130     auto operator=(const AccountActivity&) -> AccountActivity& = delete;
131     auto operator=(AccountActivity&&) -> AccountActivity& = delete;
132 };
133 }  // namespace opentxs::ui
```

7.34 AccountCurrency.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/core/UnitType.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include "opentxs/core/Types.hpp"
13 #include "opentxs/interface/ui/List.hpp"
14 #include "opentxs/interface/ui/ListRow.hpp"
15 #include "opentxs/util/Container.hpp"
16 #include "opentxs/util/SharedPimpl.hpp"
17
18 // NOLINTBEGIN(modernize-concat-nested-namespaces)
19 namespace opentxs  // NOLINT
20 {
21 // inline namespace v1
22 // {
23 namespace ui
24 {
25 class AccountTreeItem;
26 }  // namespace ui
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::ui
32 {
39 class OPENTXS_EXPORT AccountCurrency : virtual public List,
40                                        virtual public ListRow
41 {
42 public:
43     virtual auto Currency() const noexcept -> UnitType = 0;
44     virtual auto Debug() const noexcept -> UnallocatedCString = 0;
45     virtual auto First() const noexcept -> SharedPimpl<AccountTreeItem> = 0;
46     virtual auto Name() const noexcept -> UnallocatedCString = 0;
47     virtual auto Next() const noexcept -> SharedPimpl<AccountTreeItem> = 0;
```

```
48
49      ~AccountCurrency() override = default;
50
51  protected:
52      AccountCurrency() noexcept = default;
53
54  private:
55      AccountCurrency(const AccountCurrency&) = delete;
56      AccountCurrency(AccountCurrency&&) = delete;
57      auto operator=(const AccountCurrency&) -> AccountCurrency& = delete;
58      auto operator=(AccountCurrency&&) -> AccountCurrency& = delete;
59  };
60  }  // namespace opentxs::ui
```

7.35 AccountList.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class AccountList;
21 class AccountListItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
33 class OPENTXS_EXPORT AccountList : virtual public List
34 {
35 public:
37      virtual auto First() const noexcept
38          -> opentxs::SharedPimpl<opentxs::ui::AccountListItem> = 0;
40      virtual auto Next() const noexcept
41          -> opentxs::SharedPimpl<opentxs::ui::AccountListItem> = 0;
42
43      ~AccountList() override = default;
44
45  protected:
46      AccountList() noexcept = default;
47
48  private:
49      AccountList(const AccountList&) = delete;
50      AccountList(AccountList&&) = delete;
51      auto operator=(const AccountList&) -> AccountList& = delete;
52      auto operator=(AccountList&&) -> AccountList& = delete;
53  };
54  }  // namespace opentxs::ui
```

7.36 AccountListItem.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
```

```
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class AccountListItem;
22 }  // namespace ui
23
24 using OTUIAccountListItem = SharedPimpl<ui::AccountListItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
34 class OPENTXS_EXPORT AccountListItem : virtual public ListRow
35 {
36 public:
38     virtual auto AccountID() const noexcept -> UnallocatedCString = 0;
40     virtual auto Balance() const noexcept -> Amount = 0;
42     virtual auto ContractID() const noexcept -> UnallocatedCString = 0;
44     virtual auto DisplayBalance() const noexcept -> UnallocatedCString = 0;
46     virtual auto DisplayUnit() const noexcept -> UnallocatedCString = 0;
48     virtual auto Name() const noexcept -> UnallocatedCString = 0;
50     virtual auto NotaryID() const noexcept -> UnallocatedCString = 0;
52     virtual auto NotaryName() const noexcept -> UnallocatedCString = 0;
54     virtual auto Type() const noexcept -> AccountType = 0;
56     virtual auto Unit() const noexcept -> UnitType = 0;
57
58     ~AccountListItem() override = default;
59
60 protected:
61     AccountListItem() noexcept = default;
62
63 private:
64     AccountListItem(const AccountListItem&) = delete;
65     AccountListItem(AccountListItem&&) = delete;
66     auto operator=(const AccountListItem&) -> AccountListItem& = delete;
67     auto operator=(AccountListItem&&) -> AccountListItem& = delete;
68 };
69 }  // namespace opentxs::ui
```

7.37 AccountSummary.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class AccountSummary;
21 class IssuerItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
33 class OPENTXS_EXPORT AccountSummary : virtual public List
34 {
35 public:
37     virtual auto First() const noexcept
38         -> opentxs::SharedPimpl<opentxs::ui::IssuerItem> = 0;
40     virtual auto Next() const noexcept
41         -> opentxs::SharedPimpl<opentxs::ui::IssuerItem> = 0;
42
```

```
43     ~AccountSummary() override = default;
44
45 protected:
46     AccountSummary() noexcept = default;
47
48 private:
49     AccountSummary(const AccountSummary&) = delete;
50     AccountSummary(AccountSummary&&) = delete;
51     auto operator=(const AccountSummary&) -> AccountSummary& = delete;
52     auto operator=(AccountSummary&&) -> AccountSummary& = delete;
53 };
54 }  // namespace opentxs::ui
```

7.38 AccountSummaryItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class AccountSummaryItem;
22 }  // namespace ui
23
24 using OTUIAccountSummaryItem = SharedPimpl<ui::AccountSummaryItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
35 class OPENTXS_EXPORT AccountSummaryItem : virtual public ListRow
36 {
37 public:
39     virtual auto AccountID() const noexcept -> UnallocatedCString = 0;
41     virtual auto Balance() const noexcept -> Amount = 0;
43     virtual auto DisplayBalance() const noexcept -> UnallocatedCString = 0;
45     virtual auto Name() const noexcept -> UnallocatedCString = 0;
46
47     ~AccountSummaryItem() override = default;
48
49 protected:
50     AccountSummaryItem() noexcept = default;
51
52 private:
53     AccountSummaryItem(const AccountSummaryItem&) = delete;
54     AccountSummaryItem(AccountSummaryItem&&) = delete;
55     auto operator=(const AccountSummaryItem&) -> AccountSummaryItem& = delete;
56     auto operator=(AccountSummaryItem&&) -> AccountSummaryItem& = delete;
57 };
58 }  // namespace opentxs::ui
```

7.39 AccountTree.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
```

```
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace identifier
19 {
20 class Nym;
21 }  // namespace identifier
22
23 namespace ui
24 {
25 class AccountTree;
26 class AccountCurrency;
27 }  // namespace ui
28 // }  // namespace v1
29 }  // namespace opentxs
30 // NOLINTEND(modernize-concat-nested-namespaces)
31
42 namespace opentxs::ui
43 {
44 class OPENTXS_EXPORT AccountTree : virtual public List
45 {
46 public:
48     virtual auto Debug() const noexcept -> UnallocatedCString = 0;
50     virtual auto First() const noexcept
51         -> opentxs::SharedPimpl<AccountCurrency> = 0;
53     virtual auto Next() const noexcept
54         -> opentxs::SharedPimpl<AccountCurrency> = 0;
56     virtual auto Owner() const noexcept -> const identifier::Nym& = 0;
57
58     ~AccountTree() override = default;
59
60 protected:
61     AccountTree() noexcept = default;
62
63 private:
64     AccountTree(const AccountTree&) = delete;
65     AccountTree(AccountTree&&) = delete;
66     auto operator=(const AccountTree&) -> AccountTree& = delete;
67     auto operator=(AccountTree&&) -> AccountTree& = delete;
68 };
69 }  // namespace opentxs::ui
```

7.40 AccountTreeItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/core/UnitType.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include "opentxs/core/Types.hpp"
13 #include "opentxs/interface/ui/ListRow.hpp"
14 #include "opentxs/util/Container.hpp"
15 #include "opentxs/util/SharedPimpl.hpp"
16
17 // NOLINTBEGIN(modernize-concat-nested-namespaces)
18 namespace opentxs  // NOLINT
19 {
20 // inline namespace v1
21 // {
22 class Amount;
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
37 class OPENTXS_EXPORT AccountTreeItem : virtual public ListRow
38 {
39 public:
41     virtual auto AccountID() const noexcept -> UnallocatedCString = 0;
43     virtual auto Balance() const noexcept -> Amount = 0;
45     virtual auto ContractID() const noexcept -> UnallocatedCString = 0;
47     virtual auto DisplayBalance() const noexcept -> UnallocatedCString = 0;
49     virtual auto DisplayUnit() const noexcept -> UnallocatedCString = 0;
```

```
51     virtual auto Name() const noexcept -> UnallocatedCString = 0;
53     virtual auto NotaryID() const noexcept -> UnallocatedCString = 0;
55     virtual auto NotaryName() const noexcept -> UnallocatedCString = 0;
57     virtual auto Type() const noexcept -> AccountType = 0;
59     virtual auto Unit() const noexcept -> UnitType = 0;
60
61     ~AccountTreeItem() override = default;
62
63 protected:
64     AccountTreeItem() noexcept = default;
65
66 private:
67     AccountTreeItem(const AccountTreeItem&) = delete;
68     AccountTreeItem(AccountTreeItem&&) = delete;
69     auto operator=(const AccountTreeItem&) -> AccountTreeItem& = delete;
70     auto operator=(AccountTreeItem&&) -> AccountTreeItem& = delete;
71 };
72 }  // namespace opentxs::ui
```

7.41 ActivitySummary.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class ActivitySummary;
21 class ActivitySummaryItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
34 class OPENTXS_EXPORT ActivitySummary : virtual public List
35 {
36 public:
38     virtual auto First() const noexcept
39         -> opentxs::SharedPimpl<opentxs::ui::ActivitySummaryItem> = 0;
41     virtual auto Next() const noexcept
42         -> opentxs::SharedPimpl<opentxs::ui::ActivitySummaryItem> = 0;
43
44     ~ActivitySummary() override = default;
45
46 protected:
47     ActivitySummary() = default;
48
49 private:
50     ActivitySummary(const ActivitySummary&) = delete;
51     ActivitySummary(ActivitySummary&&) = delete;
52     auto operator=(const ActivitySummary&) -> ActivitySummary& = delete;
53     auto operator=(ActivitySummary&&) -> ActivitySummary& = delete;
54 };
55 }  // namespace opentxs::ui
```

7.42 ActivitySummaryItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
```

```
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11
12 #include "ListRow.hpp"
13 #include "opentxs/otx/client/Types.hpp"
14 #include "opentxs/util/Container.hpp"
15 #include "opentxs/util/SharedPimpl.hpp"
16 #include "opentxs/util/Time.hpp"
17
18 // NOLINTBEGIN(modernize-concat-nested-namespaces)
19 namespace opentxs  // NOLINT
20 {
21 // inline namespace v1
22 // {
23 namespace ui
24 {
25 class ActivitySummaryItem;
26 }  // namespace ui
27
28 using OTUIActivitySummaryItem = SharedPimpl<ui::ActivitySummaryItem>;
29 // }  // namespace v1
30 }  // namespace opentxs
31 // NOLINTEND(modernize-concat-nested-namespaces)
32
33 namespace opentxs::ui
34 {
40 class OPENTXS_EXPORT ActivitySummaryItem : virtual public ListRow
41 {
42 public:
44     virtual auto DisplayName() const noexcept -> UnallocatedCString = 0;
46     virtual auto ImageURI() const noexcept -> UnallocatedCString = 0;
48     virtual auto Text() const noexcept -> UnallocatedCString = 0;
50     virtual auto ThreadID() const noexcept -> UnallocatedCString = 0;
52     virtual auto Timestamp() const noexcept -> Time = 0;
54     virtual auto Type() const noexcept -> otx::client::StorageBox = 0;
55
56     ~ActivitySummaryItem() override = default;
57
58 protected:
59     ActivitySummaryItem() noexcept = default;
60
61 private:
62     ActivitySummaryItem(const ActivitySummaryItem&) = delete;
63     ActivitySummaryItem(ActivitySummaryItem&&) = delete;
64     auto operator=(const ActivitySummaryItem&) -> ActivitySummaryItem& = delete;
65     auto operator=(ActivitySummaryItem&&) -> ActivitySummaryItem& = delete;
66 };
67 }  // namespace opentxs::ui
```

7.43 ActivityThread.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/otx/client/Types.hpp"
12 #include "opentxs/util/Container.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace ui
21 {
22 class ActivityThread;
23 class ActivityThreadItem;
24 }  // namespace ui
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
37 class OPENTXS_EXPORT ActivityThread : virtual public List
```

```
38 {
39 public:
41     virtual auto CanMessage() const noexcept -> bool = 0;
43     virtual auto DisplayName() const noexcept -> UnallocatedCString = 0;
45     virtual auto First() const noexcept
46         -> opentxs::SharedPimpl<opentxs::ui::ActivityThreadItem> = 0;
48     virtual auto GetDraft() const noexcept -> UnallocatedCString = 0;
50     virtual auto Next() const noexcept
51         -> opentxs::SharedPimpl<opentxs::ui::ActivityThreadItem> = 0;
53     virtual auto Participants() const noexcept -> UnallocatedCString = 0;
55
63     virtual auto Pay(
64         const UnallocatedCString& amount,
65         const Identifier& sourceAccount,
66         const UnallocatedCString& memo = "",
67         const otx::client::PaymentType type =
68             otx::client::PaymentType::Cheque) const noexcept -> bool = 0;
69     virtual auto Pay(
70         const Amount amount,
71         const Identifier& sourceAccount,
72         const UnallocatedCString& memo = "",
73         const otx::client::PaymentType type =
74             otx::client::PaymentType::Cheque) const noexcept -> bool = 0;
77     virtual auto PaymentCode(const UnitType currency) const noexcept
78         -> UnallocatedCString = 0;
80     virtual auto SendDraft() const noexcept -> bool = 0;
82     virtual auto SetDraft(const UnallocatedCString& draft) const noexcept
83         -> bool = 0;
85     virtual auto ThreadID() const noexcept -> UnallocatedCString = 0;
86
87     ~ActivityThread() override = default;
88
89 protected:
90     ActivityThread() noexcept = default;
91
92 private:
93     ActivityThread(const ActivityThread&) = delete;
94     ActivityThread(ActivityThread&&) = delete;
95     auto operator=(const ActivityThread&) -> ActivityThread& = delete;
96     auto operator=(ActivityThread&&) -> ActivityThread& = delete;
97 };
98 }  // namespace opentxs::ui
```

7.44 ActivityThreadItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <cstdint>
12
13 #include "ListRow.hpp"
14 #include "opentxs/otx/client/Types.hpp"
15 #include "opentxs/util/Container.hpp"
16 #include "opentxs/util/SharedPimpl.hpp"
17
18 // NOLINTBEGIN(modernize-concat-nested-namespaces)
19 namespace opentxs  // NOLINT
20 {
21 // inline namespace v1
22 // {
23 namespace ui
24 {
25 class ActivityThreadItem;
26 }  // namespace ui
27
28 using OTUIActivityThreadItem = SharedPimpl<ui::ActivityThreadItem>;
29 // }  // namespace v1
30 }  // namespace opentxs
31 // NOLINTEND(modernize-concat-nested-namespaces)
32
33 namespace opentxs::ui
34 {
40 class OPENTXS_EXPORT ActivityThreadItem : virtual public ListRow
41 {
42 public:
44     virtual auto Amount() const noexcept -> opentxs::Amount = 0;
```

```
46      virtual auto Deposit() const noexcept -> bool = 0;
48      virtual auto DisplayAmount() const noexcept -> UnallocatedCString = 0;
50      virtual auto From() const noexcept -> UnallocatedCString = 0;
52      virtual auto Loading() const noexcept -> bool = 0;
54      virtual auto MarkRead() const noexcept -> bool = 0;
56      virtual auto Memo() const noexcept -> UnallocatedCString = 0;
58      virtual auto Outgoing() const noexcept -> bool = 0;
60      virtual auto Pending() const noexcept -> bool = 0;
62      virtual auto Text() const noexcept -> UnallocatedCString = 0;
64      virtual auto Timestamp() const noexcept -> Time = 0;
66      virtual auto Type() const noexcept -> otx::client::StorageBox = 0;
67
68      ~ActivityThreadItem() override = default;
69
70  protected:
71      ActivityThreadItem() noexcept = default;
72
73  private:
74      ActivityThreadItem(const ActivityThreadItem&) = delete;
75      ActivityThreadItem(ActivityThreadItem&&) = delete;
76      auto operator=(const ActivityThreadItem&) -> ActivityThreadItem& = delete;
77      auto operator=(ActivityThreadItem&&) -> ActivityThreadItem& = delete;
78  };
79  }  // namespace opentxs::ui
```

7.45 BalanceItem.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <chrono>
11 #include <cstdint>
12
13 #include "ListRow.hpp"
14 #include "opentxs/otx/client/Types.hpp"
15 #include "opentxs/util/Container.hpp"
16 #include "opentxs/util/SharedPimpl.hpp"
17 #include "opentxs/util/Time.hpp"
18
19 // NOLINTBEGIN(modernize-concat-nested-namespaces)
20 namespace opentxs  // NOLINT
21 {
22 // inline namespace v1
23 // {
24 namespace ui
25 {
26 class BalanceItem;
27 }  // namespace ui
28
29 using OTUIBalanceItem = SharedPimpl<ui::BalanceItem>;
30 // }  // namespace v1
31 }  // namespace opentxs
32 // NOLINTEND(modernize-concat-nested-namespaces)
33
34 namespace opentxs::ui
35 {
40 class OPENTXS_EXPORT BalanceItem : virtual public ListRow
41 {
42 public:
44      virtual auto Amount() const noexcept -> opentxs::Amount = 0;
46      virtual auto Confirmations() const noexcept -> int = 0;
48      virtual auto Contacts() const noexcept
49          -> UnallocatedVector<UnallocatedCString> = 0;
51      virtual auto DisplayAmount() const noexcept -> UnallocatedCString = 0;
53      virtual auto Memo() const noexcept -> UnallocatedCString = 0;
55      virtual auto Text() const noexcept -> UnallocatedCString = 0;
57      virtual auto Timestamp() const noexcept -> Time = 0;
59      virtual auto Type() const noexcept -> otx::client::StorageBox = 0;
61      virtual auto UUID() const noexcept -> UnallocatedCString = 0;
63      virtual auto Workflow() const noexcept -> UnallocatedCString = 0;
64
65      ~BalanceItem() override = default;
66
67  protected:
68      BalanceItem() noexcept = default;
69
70  private:
```

```
71     BalanceItem(const BalanceItem&) = delete;
72     BalanceItem(BalanceItem&&) = delete;
73     auto operator=(const BalanceItem&) -> BalanceItem& = delete;
74     auto operator=(BalanceItem&&) -> BalanceItem& = delete;
75 };
76 }  // namespace opentxs::ui
```

7.46 BlockchainAccountStatus.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/blockchain/BlockchainType.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include "opentxs/blockchain/Types.hpp"
13 #include "opentxs/interface/ui/List.hpp"
14 #include "opentxs/util/SharedPimpl.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace identifier
22 {
23 class Nym;
24 }  // namespace identifier
25
26 namespace ui
27 {
28 class BlockchainSubaccountSource;
29 }  // namespace ui
30 // }  // namespace v1
31 }  // namespace opentxs
32 // NOLINTEND(modernize-concat-nested-namespaces)
33
34 namespace opentxs::ui
35 {
40 class OPENTXS_EXPORT BlockchainAccountStatus : virtual public List
41 {
42 public:
44     virtual auto Chain() const noexcept -> blockchain::Type = 0;
46     virtual auto First() const noexcept
47         -> SharedPimpl<BlockchainSubaccountSource> = 0;
49     virtual auto Next() const noexcept
50         -> SharedPimpl<BlockchainSubaccountSource> = 0;
52     virtual auto Owner() const noexcept -> const identifier::Nym& = 0;
53
54     ~BlockchainAccountStatus() override = default;
55
56 protected:
57     BlockchainAccountStatus() noexcept = default;
58
59 private:
60     BlockchainAccountStatus(const BlockchainAccountStatus&) = delete;
61     BlockchainAccountStatus(BlockchainAccountStatus&&) = delete;
62     auto operator=(const BlockchainAccountStatus&)
63         -> BlockchainAccountStatus& = delete;
64     auto operator=(BlockchainAccountStatus&&)
65         -> BlockchainAccountStatus& = delete;
66 };
67 }  // namespace opentxs::ui
```

7.47 Blockchains.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"            // IWYU pragma: associated
```

```
9 #include "opentxs/interface/ui/Types.hpp"  // IWYU pragma: associated
10
11 #include <cstdint>
12
13 namespace opentxs::ui
14 {
15 enum class Blockchains : std::uint8_t {
16     All = 0,
17     Main = 1,
18     Test = 2,
19 };
20 }  // namespace opentxs::ui
```

7.48 BlockchainSelection.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/blockchain/Types.hpp"
11 #include "opentxs/interface/ui/List.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class BlockchainSelection;
22 class BlockchainSelectionItem;
23 }  // namespace ui
24 // }  // namespace v1
25 }  // namespace opentxs
26 // NOLINTEND(modernize-concat-nested-namespaces)
27
28 namespace opentxs::ui
29 {
34 class OPENTXS_EXPORT BlockchainSelection : virtual public List
35 {
36 public:
38     virtual auto First() const noexcept
39         -> opentxs::SharedPimpl<opentxs::ui::BlockchainSelectionItem> = 0;
41     virtual auto Next() const noexcept
42         -> opentxs::SharedPimpl<opentxs::ui::BlockchainSelectionItem> = 0;
43
45     virtual auto Disable(const blockchain::Type type) const noexcept
46         -> bool = 0;
48     virtual auto Enable(const blockchain::Type type) const noexcept -> bool = 0;
49
50     ~BlockchainSelection() override = default;
51
52 protected:
53     BlockchainSelection() noexcept = default;
54
55 private:
56     BlockchainSelection(const BlockchainSelection&) = delete;
57     BlockchainSelection(BlockchainSelection&&) = delete;
58     auto operator=(const BlockchainSelection&) -> BlockchainSelection& = delete;
59     auto operator=(BlockchainSelection&&) -> BlockchainSelection& = delete;
60 };
61 }  // namespace opentxs::ui
```

7.49 BlockchainSelectionItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
```

```
9
10 #include "opentxs/blockchain/Types.hpp"
11 #include "opentxs/interface/ui/ListRow.hpp"
12 #include "opentxs/util/Container.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace ui
21 {
22 class BlockchainSelectionItem;
23 }  // namespace ui
24
25 using OTUIBlockchainSelectionItem = SharedPimpl<ui::BlockchainSelectionItem>;
26 // }  // namespace v1
27 }  // namespace opentxs
28 // NOLINTEND(modernize-concat-nested-namespaces)
29
30 namespace opentxs::ui
31 {
35 class OPENTXS_EXPORT BlockchainSelectionItem : virtual public ListRow
36 {
37 public:
39     virtual auto Name() const noexcept -> UnallocatedCString = 0;
41     virtual auto IsEnabled() const noexcept -> bool = 0;
43     virtual auto IsTestnet() const noexcept -> bool = 0;
45     virtual auto Type() const noexcept -> blockchain::Type = 0;
46
47     ~BlockchainSelectionItem() override = default;
48
49 protected:
50     BlockchainSelectionItem() noexcept = default;
51
52 private:
53     BlockchainSelectionItem(const BlockchainSelectionItem&) = delete;
54     BlockchainSelectionItem(BlockchainSelectionItem&&) = delete;
55     auto operator=(const BlockchainSelectionItem&)
56         -> BlockchainSelectionItem& = delete;
57     auto operator=(BlockchainSelectionItem&&)
58         -> BlockchainSelectionItem& = delete;
59 };
60 }  // namespace opentxs::ui
```

7.50 BlockchainStatistics.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/blockchain/Types.hpp"
11 #include "opentxs/interface/ui/List.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class BlockchainStatistics;
22 class BlockchainStatisticsItem;
23 }  // namespace ui
24 // }  // namespace v1
25 }  // namespace opentxs
26 // NOLINTEND(modernize-concat-nested-namespaces)
27
28 namespace opentxs::ui
29 {
33 class OPENTXS_EXPORT BlockchainStatistics : virtual public List
34 {
35 public:
37     virtual auto First() const noexcept
38         -> opentxs::SharedPimpl<opentxs::ui::BlockchainStatisticsItem> = 0;
40     virtual auto Next() const noexcept
```

```
41         -> opentxs::SharedPimpl<opentxs::ui::BlockchainStatisticsItem> = 0;
42
43     ~BlockchainStatistics() override = default;
44
45 protected:
46     BlockchainStatistics() noexcept = default;
47
48 private:
49     BlockchainStatistics(const BlockchainStatistics&) = delete;
50     BlockchainStatistics(BlockchainStatistics&&) = delete;
51     auto operator=(const BlockchainStatistics&)
52         -> BlockchainStatistics& = delete;
53     auto operator=(BlockchainStatistics&&) -> BlockchainStatistics& = delete;
54 };
55 }  // namespace opentxs::ui
```

7.51 BlockchainStatisticsItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/blockchain/Types.hpp"
11 #include "opentxs/blockchain/block/Position.hpp"
12 #include "opentxs/blockchain/block/Types.hpp"
13 #include "opentxs/interface/ui/ListRow.hpp"
14 #include "opentxs/util/Container.hpp"
15 #include "opentxs/util/SharedPimpl.hpp"
16
17 // NOLINTBEGIN(modernize-concat-nested-namespaces)
18 namespace opentxs  // NOLINT
19 {
20 // inline namespace v1
21 // {
22 namespace ui
23 {
24 class BlockchainStatisticsItem;
25 }  // namespace ui
26
27 using OTUIBlockchainStatisticsItem = SharedPimpl<ui::BlockchainStatisticsItem>;
28 // }  // namespace v1
29 }  // namespace opentxs
30 // NOLINTEND(modernize-concat-nested-namespaces)
31
32 namespace opentxs::ui
33 {
38 class OPENTXS_EXPORT BlockchainStatisticsItem : virtual public ListRow
39 {
40 public:
41     using Position = blockchain::block::Height;
42
44     virtual auto ActivePeers() const noexcept -> std::size_t = 0;
46     virtual auto Balance() const noexcept -> UnallocatedCString = 0;
48     virtual auto BlockDownloadQueue() const noexcept -> std::size_t = 0;
50     virtual auto Chain() const noexcept -> blockchain::Type = 0;
52     virtual auto ConnectedPeers() const noexcept -> std::size_t = 0;
54     virtual auto Filters() const noexcept -> Position = 0;
56     virtual auto Headers() const noexcept -> Position = 0;
58     virtual auto Name() const noexcept -> UnallocatedCString = 0;
59
60     ~BlockchainStatisticsItem() override = default;
61
62 protected:
63     BlockchainStatisticsItem() noexcept = default;
64
65 private:
66     BlockchainStatisticsItem(const BlockchainStatisticsItem&) = delete;
67     BlockchainStatisticsItem(BlockchainStatisticsItem&&) = delete;
68     auto operator=(const BlockchainStatisticsItem&)
69         -> BlockchainStatisticsItem& = delete;
70     auto operator=(BlockchainStatisticsItem&&)
71         -> BlockchainStatisticsItem& = delete;
72 };
73 }  // namespace opentxs::ui
```

7.52 BlockchainSubaccount.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/interface/ui/ListRow.hpp"
12 #include "opentxs/util/Container.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace ui
21 {
22 class BlockchainSubchain;
23 }  // namespace ui
24 // }  // namespace v1
25 }  // namespace opentxs
26 // NOLINTEND(modernize-concat-nested-namespaces)
27
28 namespace opentxs::ui
29 {
30 class OPENTXS_EXPORT BlockchainSubaccount : virtual public List,
31                                             virtual public ListRow
32 {
33 public:
34     virtual auto First() const noexcept -> SharedPimpl<BlockchainSubchain> = 0;
35     virtual auto Name() const noexcept -> UnallocatedCString = 0;
36     virtual auto Next() const noexcept -> SharedPimpl<BlockchainSubchain> = 0;
37     virtual auto SubaccountID() const noexcept -> const Identifier& = 0;
38
39     ~BlockchainSubaccount() override = default;
40
41 protected:
42     BlockchainSubaccount() noexcept = default;
43
44 private:
45     BlockchainSubaccount(const BlockchainSubaccount&) = delete;
46     BlockchainSubaccount(BlockchainSubaccount&&) = delete;
47     auto operator=(const BlockchainSubaccount&)
48         -> BlockchainSubaccount& = delete;
49     auto operator=(BlockchainSubaccount&&) -> BlockchainSubaccount& = delete;
50 };
51 }  // namespace opentxs::ui
```

7.53 BlockchainSubaccountSource.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/blockchain/crypto/SubaccountType.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include "opentxs/blockchain/crypto/Types.hpp"
13 #include "opentxs/interface/ui/List.hpp"
14 #include "opentxs/interface/ui/ListRow.hpp"
15 #include "opentxs/util/Container.hpp"
16 #include "opentxs/util/SharedPimpl.hpp"
17
18 // NOLINTBEGIN(modernize-concat-nested-namespaces)
19 namespace opentxs  // NOLINT
20 {
21 // inline namespace v1
22 // {
23 namespace ui
24 {
25 class BlockchainSubaccount;
26 }  // namespace ui
27 // }  // namespace v1
```

```
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces
30
31 namespace opentxs::ui
32 {
33 class OPENTXS_EXPORT BlockchainSubaccountSource : virtual public List,
34                                                   virtual public ListRow
35 {
36 public:
37     virtual auto First() const noexcept
38         -> SharedPimpl<BlockchainSubaccount> = 0;
39     virtual auto Name() const noexcept -> UnallocatedCString = 0;
40     virtual auto Next() const noexcept -> SharedPimpl<BlockchainSubaccount> = 0;
41     virtual auto SourceID() const noexcept -> const Identifier& = 0;
42     virtual auto Type() const noexcept
43         -> blockchain::crypto::SubaccountType = 0;
44
45     ~BlockchainSubaccountSource() override = default;
46
47 protected:
48     BlockchainSubaccountSource() noexcept = default;
49
50 private:
51     BlockchainSubaccountSource(const BlockchainSubaccountSource&) = delete;
52     BlockchainSubaccountSource(BlockchainSubaccountSource&&) = delete;
53     auto operator=(const BlockchainSubaccountSource&)
54         -> BlockchainSubaccountSource& = delete;
55     auto operator=(BlockchainSubaccountSource&&)
56         -> BlockchainSubaccountSource& = delete;
57 };
58 }  // namespace opentxs::ui
```

7.54 BlockchainSubchain.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/blockchain/crypto/Subchain.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include "opentxs/blockchain/crypto/Types.hpp"
13 #include "opentxs/interface/ui/ListRow.hpp"
14 #include "opentxs/util/Container.hpp"
15 #include "opentxs/util/SharedPimpl.hpp"
16
17 namespace opentxs::ui
18 {
19 class OPENTXS_EXPORT BlockchainSubchain : virtual public ListRow
20 {
21 public:
22     virtual auto Name() const noexcept -> UnallocatedCString = 0;
23     virtual auto Progress() const noexcept -> UnallocatedCString = 0;
24     virtual auto Type() const noexcept -> blockchain::crypto::Subchain = 0;
25
26     ~BlockchainSubchain() override = default;
27
28 protected:
29     BlockchainSubchain() noexcept = default;
30
31 private:
32     BlockchainSubchain(const BlockchainSubchain&) = delete;
33     BlockchainSubchain(BlockchainSubchain&&) = delete;
34     auto operator=(const BlockchainSubchain&) -> BlockchainSubchain& = delete;
35     auto operator=(BlockchainSubchain&&) -> BlockchainSubchain& = delete;
36 };
37 }  // namespace opentxs::ui
```

7.55 Contact.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
```

```
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class Contact;
22 class ContactSection;
23 }  // namespace ui
24 // }  // namespace v1
25 }  // namespace opentxs
26 // NOLINTEND(modernize-concat-nested-namespaces)
27
28 namespace opentxs::ui
29 {
34 class OPENTXS_EXPORT Contact : virtual public List
35 {
36 public:
38     virtual auto ContactID() const noexcept -> UnallocatedCString = 0;
40     virtual auto DisplayName() const noexcept -> UnallocatedCString = 0;
42     virtual auto First() const noexcept
43         -> opentxs::SharedPimpl<opentxs::ui::ContactSection> = 0;
45     virtual auto Next() const noexcept
46         -> opentxs::SharedPimpl<opentxs::ui::ContactSection> = 0;
48     virtual auto PaymentCode() const noexcept -> UnallocatedCString = 0;
49
50     ~Contact() override = default;
51
52 protected:
53     Contact() noexcept = default;
54
55 private:
56     Contact(const Contact&) = delete;
57     Contact(Contact&&) = delete;
58     auto operator=(const Contact&) -> Contact& = delete;
59     auto operator=(Contact&&) -> Contact& = delete;
60 };
61 }  // namespace opentxs::ui
```

7.56 ContactItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class ContactItem;
22 }  // namespace ui
23
24 using OTUIContactItem = SharedPimpl<ui::ContactItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
35 class OPENTXS_EXPORT ContactItem : virtual public ListRow
36 {
37 public:
```

```
39      virtual auto ClaimID() const noexcept -> UnallocatedCString = 0;
41      virtual auto IsActive() const noexcept -> bool = 0;
43      virtual auto IsPrimary() const noexcept -> bool = 0;
45      virtual auto Value() const noexcept -> UnallocatedCString = 0;
46
47      ~ContactItem() override = default;
48
49  protected:
50      ContactItem() noexcept = default;
51
52  private:
53      ContactItem(const ContactItem&) = delete;
54      ContactItem(ContactItem&&) = delete;
55      auto operator=(const ContactItem&) -> ContactItem& = delete;
56      auto operator=(ContactItem&&) -> ContactItem& = delete;
57  };
58  }  // namespace opentxs::ui
```

7.57 ContactList.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class ContactList;
21 class ContactListItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
33 class OPENTXS_EXPORT ContactList : virtual public List
34 {
35 public:
37      virtual auto AddContact(
38          const UnallocatedCString& label,
39          const UnallocatedCString& paymentCode = "",
40          const UnallocatedCString& nymID = "") const noexcept
41          -> UnallocatedCString = 0;
43      virtual auto First() const noexcept
44          -> opentxs::SharedPimpl<opentxs::ui::ContactListItem> = 0;
46      virtual auto Next() const noexcept
47          -> opentxs::SharedPimpl<opentxs::ui::ContactListItem> = 0;
48
49      OPENTXS_NO_EXPORT ~ContactList() override = default;
50
51  protected:
52      ContactList() noexcept = default;
53
54  private:
55      ContactList(const ContactList&) = delete;
56      ContactList(ContactList&&) = delete;
57      auto operator=(const ContactList&) -> ContactList& = delete;
58      auto operator=(ContactList&&) -> ContactList& = delete;
59  };
60  }  // namespace opentxs::ui
```

7.58 ContactListItem.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
```

```
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class ContactListItem;
22 }  // namespace ui
23
24 using OTUIContactListItem = SharedPimpl<ui::ContactListItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
34 class OPENTXS_EXPORT ContactListItem : virtual public ListRow
35 {
36 public:
38     virtual auto ContactID() const noexcept -> UnallocatedCString = 0;
40     virtual auto DisplayName() const noexcept -> UnallocatedCString = 0;
42     virtual auto ImageURI() const noexcept -> UnallocatedCString = 0;
44     virtual auto Section() const noexcept -> UnallocatedCString = 0;
45
46     ~ContactListItem() override = default;
47
48 protected:
49     ContactListItem() noexcept = default;
50
51 private:
52     ContactListItem(const ContactListItem&) = delete;
53     ContactListItem(ContactListItem&&) = delete;
54     auto operator=(const ContactListItem&) -> ContactListItem& = delete;
55     auto operator=(ContactListItem&&) -> ContactListItem& = delete;
56 };
57 }  // namespace opentxs::ui
```

7.59 ContactSection.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/identity/wot/claim/Types.hpp"
11 #include "opentxs/interface/ui/List.hpp"
12 #include "opentxs/interface/ui/ListRow.hpp"
13 #include "opentxs/util/Container.hpp"
14 #include "opentxs/util/SharedPimpl.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace ui
22 {
23 class ContactSection;
24 class ContactSubsection;
25 }  // namespace ui
26
27 using OTUIContactSection = SharedPimpl<ui::ContactSection>;
28 // }  // namespace v1
29 }  // namespace opentxs
30 // NOLINTEND(modernize-concat-nested-namespaces)
31
32 namespace opentxs::ui
33 {
```

```
38 class OPENTXS_EXPORT ContactSection : virtual public List,
39                                       virtual public ListRow
40 {
41 public:
43     virtual auto Name(const UnallocatedCString& lang) const noexcept
44         -> UnallocatedCString = 0;
46     virtual auto First() const noexcept
47         -> opentxs::SharedPimpl<opentxs::ui::ContactSubsection> = 0;
49     virtual auto Next() const noexcept
50         -> opentxs::SharedPimpl<opentxs::ui::ContactSubsection> = 0;
52     virtual auto Type() const noexcept -> identity::wot::claim::SectionType = 0;
53
54     ~ContactSection() override = default;
55
56 protected:
57     ContactSection() noexcept = default;
58
59 private:
60     ContactSection(const ContactSection&) = delete;
61     ContactSection(ContactSection&&) = delete;
62     auto operator=(const ContactSection&) -> ContactSection& = delete;
63     auto operator=(ContactSection&&) -> ContactSection& = delete;
64 };
65 }  // namespace opentxs::ui
```

7.60 ContactSubsection.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/interface/ui/ListRow.hpp"
12 #include "opentxs/util/Container.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace ui
21 {
22 class ContactItem;
23 class ContactSubsection;
24 }  // namespace ui
25
26 using OTUIContactSubsection = SharedPimpl<ui::ContactSubsection>;
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::ui
32 {
37 class OPENTXS_EXPORT ContactSubsection : virtual public List,
38                                          virtual public ListRow
39 {
40 public:
42     virtual auto Name(const UnallocatedCString& lang) const noexcept
43         -> UnallocatedCString = 0;
45     virtual auto First() const noexcept
46         -> opentxs::SharedPimpl<opentxs::ui::ContactItem> = 0;
48     virtual auto Next() const noexcept
49         -> opentxs::SharedPimpl<opentxs::ui::ContactItem> = 0;
51     virtual auto Type() const noexcept -> identity::wot::claim::ClaimType = 0;
52
53     ~ContactSubsection() override = default;
54
55 protected:
56     ContactSubsection() noexcept = default;
57
58 private:
59     ContactSubsection(const ContactSubsection&) = delete;
60     ContactSubsection(ContactSubsection&&) = delete;
61     auto operator=(const ContactSubsection&) -> ContactSubsection& = delete;
62     auto operator=(ContactSubsection&&) -> ContactSubsection& = delete;
63 };
64 }  // namespace opentxs::ui
```

7.61 IssuerItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/interface/ui/ListRow.hpp"
12 #include "opentxs/util/Container.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace ui
21 {
22 class AccountSummaryItem;
23 class IssuerItem;
24 }  // namespace ui
25
26 using OTUIIssuerItem = SharedPimpl<ui::IssuerItem>;
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::ui
32 {
33 class OPENTXS_EXPORT IssuerItem : virtual public List, virtual public ListRow
34 {
35 public:
36     virtual auto ConnectionState() const noexcept -> bool = 0;
37     virtual auto Debug() const noexcept -> UnallocatedCString = 0;
38     virtual auto First() const noexcept
39         -> opentxs::SharedPimpl<opentxs::ui::AccountSummaryItem> = 0;
40     virtual auto Name() const noexcept -> UnallocatedCString = 0;
41     virtual auto Next() const noexcept
42         -> opentxs::SharedPimpl<opentxs::ui::AccountSummaryItem> = 0;
43     virtual auto Trusted() const noexcept -> bool = 0;
44
45     ~IssuerItem() override = default;
46
47 protected:
48     IssuerItem() noexcept = default;
49
50 private:
51     IssuerItem(const IssuerItem&) = delete;
52     IssuerItem(IssuerItem&&) = delete;
53     auto operator=(const IssuerItem&) -> IssuerItem& = delete;
54     auto operator=(IssuerItem&&) -> IssuerItem& = delete;
55 };
56 }  // namespace opentxs::ui
```

7.62 List.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cassert>
11 #include <limits>
12
13 #include "opentxs/interface/ui/Widget.hpp"
14
15 namespace opentxs::ui
16 {
17 class OPENTXS_EXPORT List : virtual public Widget
18 {
19 public:
20     ~List() override = default;
21
22 protected:
```

```
23     List() noexcept = default;
24
25 private:
26     List(const List&) = delete;
27     List(List&&) = delete;
28     auto operator=(const List&) -> List& = delete;
29     auto operator=(List&&) -> List& = delete;
30 };
31 }  // namespace opentxs::ui
```

7.63 ListRow.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/Widget.hpp"
11 #include "opentxs/util/Container.hpp"
12
13 namespace opentxs::ui
14 {
15 class OPENTXS_EXPORT ListRow : virtual public Widget
16 {
17 public:
18     virtual auto Last() const noexcept -> bool = 0;
19     virtual auto Valid() const noexcept -> bool = 0;
20
21     ~ListRow() override = default;
22
23 protected:
24     ListRow() noexcept = default;
25
26 private:
27     ListRow(const ListRow&) = delete;
28     ListRow(ListRow&&) = delete;
29     auto operator=(const ListRow&) -> ListRow& = delete;
30     auto operator=(ListRow&&) -> ListRow& = delete;
31 };
32 }  // namespace opentxs::ui
```

7.64 MessagableList.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class ContactListItem;
21 class MessagableList;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
34 class OPENTXS_EXPORT MessagableList : virtual public List
35 {
36 public:
38     virtual auto First() const noexcept
```

```
39          -> opentxs::SharedPimpl<opentxs::ui::ContactListItem> = 0;
41      virtual auto Next() const noexcept
42          -> opentxs::SharedPimpl<opentxs::ui::ContactListItem> = 0;
43
44      ~MessagableList() override = default;
45
46  protected:
47      MessagableList() noexcept = default;
48
49  private:
50      MessagableList(const MessagableList&) = delete;
51      MessagableList(MessagableList&&) = delete;
52      auto operator=(const MessagableList&) -> MessagableList& = delete;
53      auto operator=(MessagableList&&) -> MessagableList& = delete;
54  };
55  }  // namespace opentxs::ui
```

7.65 NymList.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class NymList;
21 class NymListItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
33 class OPENTXS_EXPORT NymList : virtual public List
34 {
35 public:
37      virtual auto First() const noexcept
38          -> opentxs::SharedPimpl<opentxs::ui::NymListItem> = 0;
40      virtual auto Next() const noexcept
41          -> opentxs::SharedPimpl<opentxs::ui::NymListItem> = 0;
42
43      ~NymList() override = default;
44
45  protected:
46      NymList() noexcept = default;
47
48  private:
49      NymList(const NymList&) = delete;
50      NymList(NymList&&) = delete;
51      auto operator=(const NymList&) -> NymList& = delete;
52      auto operator=(NymList&&) -> NymList& = delete;
53  };
54  }  // namespace opentxs::ui
```

7.66 NymListItem.hpp

```
1  // Copyright (c) 2010-2022 The Open-Transactions developers
2  // This Source Code Form is subject to the terms of the Mozilla Public
3  // License, v. 2.0. If a copy of the MPL was not distributed with this
4  // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6  #pragma once
7
8  #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
```

```
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class NymListItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
32 class OPENTXS_EXPORT NymListItem : virtual public ListRow
33 {
34 public:
36     virtual auto Name() const noexcept -> UnallocatedCString = 0;
38     virtual auto NymID() const noexcept -> UnallocatedCString = 0;
39
40     ~NymListItem() override = default;
41
42 protected:
43     NymListItem() noexcept = default;
44
45 private:
46     NymListItem(const NymListItem&) = delete;
47     NymListItem(NymListItem&&) = delete;
48     auto operator=(const NymListItem&) -> NymListItem& = delete;
49     auto operator=(NymListItem&&) -> NymListItem& = delete;
50 };
51 }  // namespace opentxs::ui
```

7.67 PayableList.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
20 class PayableList;
21 class PayableListItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
34 class OPENTXS_EXPORT PayableList : virtual public List
35 {
36 public:
38     virtual auto First() const noexcept
39         -> opentxs::SharedPimpl<opentxs::ui::PayableListItem> = 0;
41     virtual auto Next() const noexcept
42         -> opentxs::SharedPimpl<opentxs::ui::PayableListItem> = 0;
43
44     ~PayableList() override = default;
45
46 protected:
47     PayableList() noexcept = default;
48
49 private:
50     PayableList(const PayableList&) = delete;
```

```
51     PayableList(PayableList&&) = delete;
52     auto operator=(const PayableList&) -> PayableList& = delete;
53     auto operator=(PayableList&&) -> PayableList& = delete;
54 };
55 }  // namespace opentxs::ui
```

7.68 PayableListItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "ContactListItem.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class PayableListItem;
22 }  // namespace ui
23
24 using OTUIPayableListItem = SharedPimpl<ui::PayableListItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
36 class OPENTXS_EXPORT PayableListItem : virtual public ContactListItem
37 {
38 public:
40     virtual auto PaymentCode() const noexcept -> UnallocatedCString = 0;
41
42     ~PayableListItem() override = default;
43
44 protected:
45     PayableListItem() noexcept = default;
46
47 private:
48     PayableListItem(const PayableListItem&) = delete;
49     PayableListItem(PayableListItem&&) = delete;
50     auto operator=(const PayableListItem&) -> PayableListItem& = delete;
51     auto operator=(PayableListItem&&) -> PayableListItem& = delete;
52 };
53 }  // namespace opentxs::ui
```

7.69 Profile.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <algorithm>
11 #include <tuple>
12
13 #include "opentxs/identity/wot/claim/Types.hpp"
14 #include "opentxs/interface/ui/List.hpp"
15 #include "opentxs/util/Container.hpp"
16 #include "opentxs/util/SharedPimpl.hpp"
17
18 // NOLINTBEGIN(modernize-concat-nested-namespaces)
19 namespace opentxs  // NOLINT
20 {
21 // inline namespace v1
```

```
// {
namespace ui
{
class Profile;
class ProfileSection;
}  // namespace ui
// }  // namespace v1
}  // namespace opentxs
// NOLINTEND(modernize-concat-nested-namespaces)

namespace opentxs::ui
{
class OPENTXS_EXPORT Profile : virtual public List
{
public:
    using ItemType =
        std::pair<identity::wot::claim::ClaimType, UnallocatedCString>;
    using ItemTypeList = UnallocatedVector<ItemType>;
    using SectionType =
        std::pair<identity::wot::claim::SectionType, UnallocatedCString>;
    using SectionTypeList = UnallocatedVector<SectionType>;

    virtual auto AddClaim(
        const identity::wot::claim::SectionType section,
        const identity::wot::claim::ClaimType type,
        const UnallocatedCString& value,
        const bool primary,
        const bool active) const noexcept -> bool = 0;
    virtual auto AllowedItems(
        const identity::wot::claim::SectionType section,
        const UnallocatedCString& lang) const noexcept -> ItemTypeList = 0;
    virtual auto AllowedSections(const UnallocatedCString& lang) const noexcept
        -> SectionTypeList = 0;
    virtual auto Delete(
        const int section,
        const int type,
        const UnallocatedCString& claimID) const noexcept -> bool = 0;
    virtual auto DisplayName() const noexcept -> UnallocatedCString = 0;
    virtual auto First() const noexcept
        -> opentxs::SharedPimpl<opentxs::ui::ProfileSection> = 0;
    virtual auto ID() const noexcept -> UnallocatedCString = 0;
    virtual auto Next() const noexcept
        -> opentxs::SharedPimpl<opentxs::ui::ProfileSection> = 0;
    virtual auto PaymentCode() const noexcept -> UnallocatedCString = 0;
    virtual auto SetActive(
        const int section,
        const int type,
        const UnallocatedCString& claimID,
        const bool active) const noexcept -> bool = 0;
    virtual auto SetPrimary(
        const int section,
        const int type,
        const UnallocatedCString& claimID,
        const bool primary) const noexcept -> bool = 0;
    virtual auto SetValue(
        const int section,
        const int type,
        const UnallocatedCString& claimID,
        const UnallocatedCString& value) const noexcept -> bool = 0;

    ~Profile() override = default;

protected:
    Profile() noexcept = default;

private:
    Profile(const Profile&) = delete;
    Profile(Profile&&) = delete;
    auto operator=(const Profile&) -> Profile& = delete;
    auto operator=(Profile&&) -> Profile& = delete;
};
}  // namespace opentxs::ui
```

7.70 ProfileItem.hpp

```
// Copyright (c) 2010-2022 The Open-Transactions developers
// This Source Code Form is subject to the terms of the Mozilla Public
// License, v. 2.0. If a copy of the MPL was not distributed with this
// file, You can obtain one at http://mozilla.org/MPL/2.0/.

#pragma once

#include "opentxs/Version.hpp"  // IWYU pragma: associated
```

```
9
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class ProfileItem;
22 }  // namespace ui
23
24 using OTUIProfileItem = SharedPimpl<ui::ProfileItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
35 class OPENTXS_EXPORT ProfileItem : virtual public ListRow
36 {
37 public:
39     virtual auto ClaimID() const noexcept -> UnallocatedCString = 0;
41     virtual auto Delete() const noexcept -> bool = 0;
43     virtual auto IsActive() const noexcept -> bool = 0;
45     virtual auto IsPrimary() const noexcept -> bool = 0;
47     virtual auto SetActive(const bool& active) const noexcept -> bool = 0;
49     virtual auto SetPrimary(const bool& primary) const noexcept -> bool = 0;
50     // Sets the value of this claim.
51     virtual auto SetValue(const UnallocatedCString& value) const noexcept
52         -> bool = 0;
54     virtual auto Value() const noexcept -> UnallocatedCString = 0;
55
56     ~ProfileItem() override = default;
57
58 protected:
59     ProfileItem() noexcept = default;
60
61 private:
62     ProfileItem(const ProfileItem&) = delete;
63     ProfileItem(ProfileItem&&) = delete;
64     auto operator=(const ProfileItem&) -> ProfileItem& = delete;
65     auto operator=(ProfileItem&&) -> ProfileItem& = delete;
66 };
67 }  // namespace opentxs::ui
```

7.71 ProfileSection.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/identity/wot/claim/Types.hpp"
11 #include "opentxs/interface/ui/List.hpp"
12 #include "opentxs/interface/ui/ListRow.hpp"
13 #include "opentxs/util/Container.hpp"
14 #include "opentxs/util/SharedPimpl.hpp"
15
16 // NOLINTBEGIN(modernize-concat-nested-namespaces)
17 namespace opentxs  // NOLINT
18 {
19 // inline namespace v1
20 // {
21 namespace ui
22 {
23 class ProfileSection;
24 class ProfileSubsection;
25 }  // namespace ui
26
27 using OTUIProfileSection = SharedPimpl<ui::ProfileSection>;
28 // }  // namespace v1
29 }  // namespace opentxs
30 // NOLINTEND(modernize-concat-nested-namespaces)
31
32 namespace opentxs::ui
```

```
33 {
38 class OPENTXS_EXPORT ProfileSection : virtual public List,
39                                         virtual public ListRow
40 {
41 public:
42     using ItemType =
43         std::pair<identity::wot::claim::ClaimType, UnallocatedCString>;
44     using ItemTypeList = UnallocatedVector<ItemType>;
45
47     static auto AllowedItems(
48         const identity::wot::claim::SectionType section,
49         const UnallocatedCString& lang) noexcept -> ItemTypeList;
50
52     virtual auto AddClaim(
53         const identity::wot::claim::ClaimType type,
54         const UnallocatedCString& value,
55         const bool primary,
56         const bool active) const noexcept -> bool = 0;
58     virtual auto Delete(const int type, const UnallocatedCString& claimID)
59         const noexcept -> bool = 0;
61     virtual auto Items(const UnallocatedCString& lang) const noexcept
62         -> ItemTypeList = 0;
64     virtual auto Name(const UnallocatedCString& lang) const noexcept
65         -> UnallocatedCString = 0;
67     virtual auto First() const noexcept
68         -> opentxs::SharedPimpl<opentxs::ui::ProfileSubsection> = 0;
70     virtual auto Next() const noexcept
71         -> opentxs::SharedPimpl<opentxs::ui::ProfileSubsection> = 0;
73     virtual auto SetActive(
74         const int type,
75         const UnallocatedCString& claimID,
76         const bool active) const noexcept -> bool = 0;
78     virtual auto SetPrimary(
79         const int type,
80         const UnallocatedCString& claimID,
81         const bool primary) const noexcept -> bool = 0;
83     virtual auto SetValue(
84         const int type,
85         const UnallocatedCString& claimID,
86         const UnallocatedCString& value) const noexcept -> bool = 0;
88     virtual auto Type() const noexcept -> identity::wot::claim::SectionType = 0;
89
90     ~ProfileSection() override = default;
91
92 protected:
93     ProfileSection() noexcept = default;
94
95 private:
96     ProfileSection(const ProfileSection&) = delete;
97     ProfileSection(ProfileSection&&) = delete;
98     auto operator=(const ProfileSection&) -> ProfileSection& = delete;
99     auto operator=(ProfileSection&&) -> ProfileSection& = delete;
100 };
101 }  // namespace opentxs::ui
```

7.72 ProfileSubsection.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/interface/ui/ListRow.hpp"
12 #include "opentxs/util/Container.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace ui
21 {
22 class ProfileItem;
23 class ProfileSubsection;
24 }  // namespace ui
25
26 using OTUIProfileSubsection = SharedPimpl<ui::ProfileSubsection>;
```

```
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::ui
32 {
37 class OPENTXS_EXPORT ProfileSubsection : virtual public List,
38                                           virtual public ListRow
39 {
40 public:
42     virtual auto AddItem(
43         const UnallocatedCString& value,
44         const bool primary,
45         const bool active) const noexcept -> bool = 0;
47     virtual auto Delete(const UnallocatedCString& claimID) const noexcept
48         -> bool = 0;
50     virtual auto First() const noexcept
51         -> opentxs::SharedPimpl<opentxs::ui::ProfileItem> = 0;
53     virtual auto Name(const UnallocatedCString& lang) const noexcept
54         -> UnallocatedCString = 0;
56     virtual auto Next() const noexcept
57         -> opentxs::SharedPimpl<opentxs::ui::ProfileItem> = 0;
59     virtual auto SetActive(const UnallocatedCString& claimID, const bool active)
60         const noexcept -> bool = 0;
62     virtual auto SetPrimary(
63         const UnallocatedCString& claimID,
64         const bool primary) const noexcept -> bool = 0;
66     virtual auto SetValue(
67         const UnallocatedCString& claimID,
68         const UnallocatedCString& value) const noexcept -> bool = 0;
70     virtual auto Type() const noexcept -> identity::wot::claim::ClaimType = 0;
71
72     ~ProfileSubsection() override = default;
73
74 protected:
75     ProfileSubsection() noexcept = default;
76
77 private:
78     ProfileSubsection(const ProfileSubsection&) = delete;
79     ProfileSubsection(ProfileSubsection&&) = delete;
80     auto operator=(const ProfileSubsection&) -> ProfileSubsection& = delete;
81     auto operator=(ProfileSubsection&&) -> ProfileSubsection& = delete;
82 };
83 }  // namespace opentxs::ui
```

7.73 SeedTree.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/core/identifier/Generic.hpp"
11 #include "opentxs/core/identifier/Nym.hpp"
12 #include "opentxs/interface/ui/List.hpp"
13 #include "opentxs/util/SharedPimpl.hpp"
14
15 // NOLINTBEGIN(modernize-concat-nested-namespaces)
16 namespace opentxs  // NOLINT
17 {
18 // inline namespace v1
19 // {
20 namespace identifier
21 {
22 class Nym;
23 }  // namespace identifier
24
25 namespace ui
26 {
27 class SeedTree;
28 class SeedTreeItem;
29 }  // namespace ui
30 // }  // namespace v1
31 }  // namespace opentxs
32 // NOLINTEND(modernize-concat-nested-namespaces)
33
34 namespace opentxs::ui
35 {
39 class OPENTXS_EXPORT SeedTree : virtual public List
```

```
40 {
41 public:
43     virtual auto Debug() const noexcept -> UnallocatedCString = 0;
45     virtual auto DefaultNym() const noexcept -> OTNymID = 0;
47     virtual auto DefaultSeed() const noexcept -> OTIdentifier = 0;
49     virtual auto First() const noexcept
50         -> opentxs::SharedPimpl<SeedTreeItem> = 0;
52     virtual auto Next() const noexcept
53         -> opentxs::SharedPimpl<SeedTreeItem> = 0;
54
55     ~SeedTree() override = default;
56
57 protected:
58     SeedTree() noexcept = default;
59
60 private:
61     SeedTree(const SeedTree&) = delete;
62     SeedTree(SeedTree&&) = delete;
63     auto operator=(const SeedTree&) -> SeedTree& = delete;
64     auto operator=(SeedTree&&) -> SeedTree& = delete;
65 };
66 }  // namespace opentxs::ui
```

7.74 SeedTreeItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 // IWYU pragma: no_include "opentxs/crypto/SeedStyle.hpp"
7
8 #pragma once
9
10 #include "opentxs/Version.hpp"  // IWYU pragma: associated
11
12 #include "opentxs/crypto/Types.hpp"
13 #include "opentxs/interface/ui/List.hpp"
14 #include "opentxs/interface/ui/ListRow.hpp"
15 #include "opentxs/util/Container.hpp"
16 #include "opentxs/util/SharedPimpl.hpp"
17
18 // NOLINTBEGIN(modernize-concat-nested-namespaces)
19 namespace opentxs  // NOLINT
20 {
21 // inline namespace v1
22 // {
23 namespace ui
24 {
25 class SeedTreeNym;
26 }  // namespace ui
27 // }  // namespace v1
28 }  // namespace opentxs
29 // NOLINTEND(modernize-concat-nested-namespaces)
30
31 namespace opentxs::ui
32 {
37 class OPENTXS_EXPORT SeedTreeItem : virtual public List, virtual public ListRow
38 {
39 public:
41     virtual auto Debug() const noexcept -> UnallocatedCString = 0;
43     virtual auto First() const noexcept -> SharedPimpl<SeedTreeNym> = 0;
45     virtual auto Name() const noexcept -> UnallocatedCString = 0;
47     virtual auto Next() const noexcept -> SharedPimpl<SeedTreeNym> = 0;
49     virtual auto SeedID() const noexcept -> UnallocatedCString = 0;
51     virtual auto Type() const noexcept -> crypto::SeedStyle = 0;
52
53     ~SeedTreeItem() override = default;
54
55 protected:
56     SeedTreeItem() noexcept = default;
57
58 private:
59     SeedTreeItem(const SeedTreeItem&) = delete;
60     SeedTreeItem(SeedTreeItem&&) = delete;
61     auto operator=(const SeedTreeItem&) -> SeedTreeItem& = delete;
62     auto operator=(SeedTreeItem&&) -> SeedTreeItem& = delete;
63 };
64 }  // namespace opentxs::ui
```

7.75 SeedTreeNym.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstddef>
11
12 #include "opentxs/interface/ui/ListRow.hpp"
13 #include "opentxs/util/Container.hpp"
14 #include "opentxs/util/SharedPimpl.hpp"
15
16 namespace opentxs::ui
17 {
22 class OPENTXS_EXPORT SeedTreeNym : virtual public ListRow
23 {
24 public:
26     virtual auto Index() const noexcept -> std::size_t = 0;
28     virtual auto Name() const noexcept -> UnallocatedCString = 0;
30     virtual auto NymID() const noexcept -> UnallocatedCString = 0;
31
32     ~SeedTreeNym() override = default;
33
34 protected:
35     SeedTreeNym() noexcept = default;
36
37 private:
38     SeedTreeNym(const SeedTreeNym&) = delete;
39     SeedTreeNym(SeedTreeNym&&) = delete;
40     auto operator=(const SeedTreeNym&) -> SeedTreeNym& = delete;
41     auto operator=(SeedTreeNym&&) -> SeedTreeNym& = delete;
42 };
43 }  // namespace opentxs::ui
```

7.76 Types.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include <cstddef>
11 #include <cstdint>
12
13 namespace opentxs::ui
14 {
15 enum class Blockchains : std::uint8_t;
16 }  // namespace opentxs::ui
```

7.77 UnitList.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/interface/ui/List.hpp"
11 #include "opentxs/util/SharedPimpl.hpp"
12
13 // NOLINTBEGIN(modernize-concat-nested-namespaces)
14 namespace opentxs  // NOLINT
15 {
16 // inline namespace v1
17 // {
18 namespace ui
19 {
```

```
20 class UnitList;
21 class UnitListItem;
22 }  // namespace ui
23 // }  // namespace v1
24 }  // namespace opentxs
25 // NOLINTEND(modernize-concat-nested-namespaces)
26
27 namespace opentxs::ui
28 {
32 class OPENTXS_EXPORT UnitList : virtual public List
33 {
34 public:
36     virtual auto First() const noexcept
37         -> opentxs::SharedPimpl<opentxs::ui::UnitListItem> = 0;
39     virtual auto Next() const noexcept
40         -> opentxs::SharedPimpl<opentxs::ui::UnitListItem> = 0;
41
42     ~UnitList() override = default;
43
44 protected:
45     UnitList() noexcept = default;
46
47 private:
48     UnitList(const UnitList&) = delete;
49     UnitList(UnitList&&) = delete;
50     auto operator=(const UnitList&) -> UnitList& = delete;
51     auto operator=(UnitList&&) -> UnitList& = delete;
52 };
53 }  // namespace opentxs::ui
```

7.78 UnitListItem.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "ListRow.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/SharedPimpl.hpp"
13
14 // NOLINTBEGIN(modernize-concat-nested-namespaces)
15 namespace opentxs  // NOLINT
16 {
17 // inline namespace v1
18 // {
19 namespace ui
20 {
21 class UnitListItem;
22 }  // namespace ui
23
24 using OTUIUnitListItem = SharedPimpl<ui::UnitListItem>;
25 // }  // namespace v1
26 }  // namespace opentxs
27 // NOLINTEND(modernize-concat-nested-namespaces)
28
29 namespace opentxs::ui
30 {
34 class OPENTXS_EXPORT UnitListItem : virtual public ListRow
35 {
36 public:
38     virtual auto Name() const noexcept -> UnallocatedCString = 0;
40     virtual auto Unit() const noexcept -> UnitType = 0;
41
42     ~UnitListItem() override = default;
43
44 protected:
45     UnitListItem() noexcept = default;
46
47 private:
48     UnitListItem(const UnitListItem&) = delete;
49     UnitListItem(UnitListItem&&) = delete;
50     auto operator=(const UnitListItem&) -> UnitListItem& = delete;
51     auto operator=(UnitListItem&&) -> UnitListItem& = delete;
52 };
53 }  // namespace opentxs::ui
```

7.79 Widget.hpp

```
1 // Copyright (c) 2010-2022 The Open-Transactions developers
2 // This Source Code Form is subject to the terms of the Mozilla Public
3 // License, v. 2.0. If a copy of the MPL was not distributed with this
4 // file, You can obtain one at http://mozilla.org/MPL/2.0/.
5
6 #pragma once
7
8 #include "opentxs/Version.hpp"  // IWYU pragma: associated
9
10 #include "opentxs/core/identifier/Generic.hpp"
11 #include "opentxs/util/Container.hpp"
12 #include "opentxs/util/Types.hpp"
13
14 namespace opentxs::ui
15 {
16 class OPENTXS_EXPORT Widget
17 {
18 public:
19     virtual void ClearCallbacks() const noexcept = 0;
20     virtual void SetCallback(SimpleCallback cb) const noexcept = 0;
21     virtual auto WidgetID() const noexcept -> OTIdentifier = 0;
22
23     virtual ~Widget() = default;
24
25 protected:
26     Widget() noexcept = default;
27
28 private:
29     Widget(const Widget&) = delete;
30     Widget(Widget&&) = delete;
31     auto operator=(const Widget&) -> Widget& = delete;
32     auto operator=(Widget&&) -> Widget& = delete;
33 };
34 }  // namespace opentxs::ui
```

Index