

libquentier

0.5.0

Generated by Doxygen 1.9.1



<b>1 libquentier</b>	<b>1</b>
1.1 What's this	1
1.1.1 WARNING: libquentier is in alpha state right now, neither API nor ABI can be considered stable yet!	1
1.2 How to build/install	1
1.3 How to contribute	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 Class Documentation</b>	<b>9</b>
4.1 quentier::Account Class Reference	9
4.1.1 Detailed Description	10
4.1.2 Member Function Documentation	11
4.1.2.1 displayName()	11
4.1.2.2 evernoteAccountType()	11
4.1.2.3 evernoteHost()	11
4.1.2.4 id()	11
4.1.2.5 isEmpty()	12
4.1.2.6 name()	12
4.1.2.7 setDisplayName()	12
4.1.2.8 shardId()	12
4.1.2.9 type()	12
4.2 quentier::ApplicationSettings Class Reference	13
4.2.1 Detailed Description	14
4.2.2 Constructor & Destructor Documentation	14
4.2.2.1 ApplicationSettings() [1/3]	14
4.2.2.2 ApplicationSettings() [2/3]	14
4.2.2.3 ApplicationSettings() [3/3]	15
4.2.2.4 ~ApplicationSettings()	15
4.2.3 Member Function Documentation	15
4.2.3.1 beginGroup() [1/2]	15
4.2.3.2 beginGroup() [2/2]	16
4.2.3.3 beginReadArray() [1/2]	16
4.2.3.4 beginReadArray() [2/2]	16
4.2.3.5 beginWriteArray() [1/2]	17
4.2.3.6 beginWriteArray() [2/2]	17
4.2.3.7 contains() [1/2]	18
4.2.3.8 contains() [2/2]	18
4.2.3.9 remove() [1/2]	18

4.2.3.10 remove() [2/2]	19
4.2.3.11 setValue() [1/2]	19
4.2.3.12 setValue() [2/2]	19
4.2.3.13 value() [1/2]	20
4.2.3.14 value() [2/2]	20
4.3 quantier::ApplicationSettingsInitializationException Class Reference	21
4.3.1 Detailed Description	22
4.4 quantier::ApplicationSettings::ArrayCloser Struct Reference	22
4.4.1 Detailed Description	22
4.5 quantier::AuthenticationManager Class Reference	23
4.5.1 Detailed Description	24
4.6 quantier::ResourceRecognitionIndexItem::BarcodeItem Struct Reference	24
4.7 quantier::DatabaseLockedException Class Reference	24
4.8 quantier::DatabaseLockFailedException Class Reference	25
4.9 quantier::DatabaseOpeningException Class Reference	26
4.10 quantier::DatabaseRequestException Class Reference	27
4.10.1 Detailed Description	29
4.11 quantier::DateTimePrint Class Reference	29
4.11.1 Detailed Description	29
4.11.2 Member Enumeration Documentation	29
4.11.2.1 Option	29
4.12 quantier::DecryptedTextManager Class Reference	29
4.13 quantier::DefaultLocalStorageCacheExpiryChecker Class Reference	30
4.13.1 Detailed Description	31
4.13.2 Member Function Documentation	32
4.13.2.1 checkLinkedNotebooks()	32
4.13.2.2 checkNotebooks()	32
4.13.2.3 checkNotes()	32
4.13.2.4 checkResources()	33
4.13.2.5 checkSavedSearches()	33
4.13.2.6 checkTags()	33
4.13.2.7 clone()	33
4.14 quantier::EmptyDataElementException Class Reference	34
4.15 quantier::EncryptionManager Class Reference	35
4.15.1 Detailed Description	36
4.16 quantier::ENMLConverter Class Reference	36
4.16.1 Detailed Description	37
4.16.2 Member Function Documentation	37
4.16.2.1 cleanupExternalHtml()	37
4.16.2.2 exportNotesToEnex()	38
4.16.2.3 htmlToQTextDocument()	38
4.16.2.4 importEnex()	39

4.17 <a href="#">quentier::ErrorString Class Reference</a>	39
4.17.1 Detailed Description	41
4.18 <a href="#">quentier::EventLoopWithExitStatus Class Reference</a>	41
4.19 <a href="#">quentier::FileCopier Class Reference</a>	42
4.20 <a href="#">quentier::FileIOProcessorAsync Class Reference</a>	44
4.20.1 Detailed Description	45
4.20.2 Member Function Documentation	45
4.20.2.1 onReadFileRequest	45
4.20.2.2 onWriteFileRequest	45
4.20.2.3 readFileRequestProcessed	46
4.20.2.4 setIdleTimePeriod()	46
4.20.2.5 writeFileRequestProcessed	47
4.21 <a href="#">quentier::FileSystemWatcher Class Reference</a>	47
4.22 <a href="#">quentier::ApplicationSettings::GroupCloser Struct Reference</a>	48
4.22.1 Detailed Description	49
4.23 <a href="#">quentier::HTMLCleaner Class Reference</a>	49
4.24 <a href="#">quentier::IAuthenticationManager Class Reference</a>	49
4.25 <a href="#">quentier::IFavoritableDataElement Class Reference</a>	50
4.25.1 Detailed Description	51
4.26 <a href="#">quentier::IKeychainService Class Reference</a>	51
4.26.1 Detailed Description	53
4.26.2 Member Enumeration Documentation	53
4.26.2.1 ErrorCode	53
4.26.3 Member Function Documentation	53
4.26.3.1 deletePasswordJobFinished	53
4.26.3.2 readPasswordJobFinished	55
4.26.3.3 startDeletePasswordJob()	55
4.26.3.4 startReadPasswordJob()	56
4.26.3.5 startWritePasswordJob()	56
4.26.3.6 writePasswordJobFinished	56
4.27 <a href="#">quentier::ILocalStorageCacheExpiryChecker Class Reference</a>	57
4.27.1 Detailed Description	58
4.27.2 Member Function Documentation	59
4.27.2.1 checkLinkedNotebooks()	59
4.27.2.2 checkNotebooks()	59
4.27.2.3 checkNotes()	59
4.27.2.4 checkResources()	59
4.27.2.5 checkSavedSearches()	60
4.27.2.6 checkTags()	60
4.27.2.7 clone()	60
4.28 <a href="#">quentier::ILocalStorageDataElement Class Reference</a>	60
4.29 <a href="#">quentier::ILocalStoragePatch Class Reference</a>	61

4.29.1 Detailed Description	62
4.29.2 Member Function Documentation	62
4.29.2.1 apply()	62
4.29.2.2 backupLocalStorage()	63
4.29.2.3 backupProgress	63
4.29.2.4 fromVersion()	63
4.29.2.5 patchLongDescription()	63
4.29.2.6 patchShortDescription()	64
4.29.2.7 progress	64
4.29.2.8 removeLocalStorageBackup()	64
4.29.2.9 restoreBackupProgress	64
4.29.2.10 restoreLocalStorageFromBackup()	65
4.29.2.11 toVersion()	65
4.30 quotienter::INoteEditorBackend Class Reference	66
4.31 quotienter::INoteStore Class Reference	69
4.31.1 Detailed Description	70
4.31.2 Member Function Documentation	71
4.31.2.1 authenticateToSharedNotebook()	71
4.31.2.2 createNote()	71
4.31.2.3 createNotebook()	72
4.31.2.4 createSavedSearch()	72
4.31.2.5 createTag()	73
4.31.2.6 getLinkedNotebookSyncChunk()	73
4.31.2.7 getLinkedNotebookSyncState()	74
4.31.2.8 getNote()	75
4.31.2.9 getNoteAsync()	75
4.31.2.10 getResource()	76
4.31.2.11 getResourceAsync()	77
4.31.2.12 getSyncChunk()	78
4.31.2.13 getSyncState()	78
4.31.2.14 noteStoreUrl()	79
4.31.2.15 setAuthData()	79
4.31.2.16 setNoteStoreUrl()	79
4.31.2.17 stop()	79
4.31.2.18 updateNote()	79
4.31.2.19 updateNotebook()	80
4.31.2.20 updateSavedSearch()	80
4.31.2.21 updateTag()	81
4.32 quotienter::INoteStoreDataElement Class Reference	81
4.33 quotienter::IQuentierException Class Reference	82
4.33.1 Detailed Description	84
4.34 quotienter::ISyncStateStorage::ISyncState Class Reference	84

4.34.1 Detailed Description	85
4.35 <code>quentier::ISyncStateStorage</code> Class Reference	85
4.35.1 Detailed Description	86
4.35.2 Member Function Documentation	86
4.35.2.1 <code>notifySyncStateUpdated</code>	87
4.36 <code>quentier::IUserStore</code> Class Reference	87
4.36.1 Detailed Description	87
4.36.2 Member Function Documentation	87
4.36.2.1 <code>checkVersion()</code>	87
4.36.2.2 <code>getAccountLimits()</code>	88
4.36.2.3 <code>getUser()</code>	88
4.36.2.4 <code>setAuthData()</code>	89
4.37 <code>quentier::LimitedStack&lt; T &gt;</code> Class Template Reference	89
4.37.1 Detailed Description	90
4.38 <code>quentier::LinkedNotebook</code> Class Reference	90
4.39 <code>quentier::LocalStorageCacheManager</code> Class Reference	92
4.40 <code>quentier::LocalStorageCacheManagerException</code> Class Reference	94
4.41 <code>quentier::LocalStorageManager</code> Class Reference	95
4.41.1 Member Enumeration Documentation	101
4.41.1.1 <code>GetNoteOption</code>	101
4.41.1.2 <code>GetResourceOption</code>	102
4.41.1.3 <code>ListObjectsOption</code>	102
4.41.1.4 <code>StartupOption</code>	102
4.41.1.5 <code>UpdateNoteOption</code>	103
4.41.2 Constructor & Destructor Documentation	103
4.41.2.1 <code>LocalStorageManager()</code>	103
4.41.3 Member Function Documentation	103
4.41.3.1 <code>accountHighUsn()</code>	104
4.41.3.2 <code>addEnResource()</code>	104
4.41.3.3 <code>addLinkedNotebook()</code>	104
4.41.3.4 <code>addNote()</code>	105
4.41.3.5 <code>addNotebook()</code>	105
4.41.3.6 <code>addSavedSearch()</code>	106
4.41.3.7 <code>addTag()</code>	106
4.41.3.8 <code>addUser()</code>	107
4.41.3.9 <code>deleteUser()</code>	107
4.41.3.10 <code>enResourceCount()</code>	107
4.41.3.11 <code>expungeEnResource()</code>	108
4.41.3.12 <code>expungeLinkedNotebook()</code>	108
4.41.3.13 <code>expungeNote()</code>	109
4.41.3.14 <code>expungeNotebook()</code>	109
4.41.3.15 <code>expungeNotelessTagsFromLinkedNotebooks()</code>	110

4.41.3.16 expungeSavedSearch()	110
4.41.3.17 expungeTag()	110
4.41.3.18 expungeUser()	111
4.41.3.19 findDefaultNotebook()	111
4.41.3.20 findDefaultOrLastUsedNotebook()	112
4.41.3.21 findEnResource()	112
4.41.3.22 findLastUsedNotebook()	113
4.41.3.23 findLinkedNotebook()	113
4.41.3.24 findNote()	114
4.41.3.25 findNotebook()	114
4.41.3.26 findNoteLocalUidsWithSearchQuery()	115
4.41.3.27 findNotesWithSearchQuery()	115
4.41.3.28 findSavedSearch()	115
4.41.3.29 findTag()	116
4.41.3.30 findUser()	116
4.41.3.31 highestSupportedLocalStorageVersion()	117
4.41.3.32 isLocalStorageVersionTooHigh()	117
4.41.3.33 linkedNotebookCount()	118
4.41.3.34 listAllLinkedNotebooks()	118
4.41.3.35 listAllNotebooks()	118
4.41.3.36 listAllSavedSearches()	119
4.41.3.37 listAllSharedNotebooks()	120
4.41.3.38 listAllTags()	120
4.41.3.39 listAllTagsPerNote()	121
4.41.3.40 listLinkedNotebooks()	121
4.41.3.41 listNotebooks()	122
4.41.3.42 listNotes()	123
4.41.3.43 listNotesByLocalUids()	123
4.41.3.44 listNotesPerNotebook()	124
4.41.3.45 listNotesPerNotebooksAndTags()	125
4.41.3.46 listNotesPerTag()	126
4.41.3.47 listSavedSearches()	126
4.41.3.48 listSharedNotebooksPerNotebookGuid()	127
4.41.3.49 listTags()	127
4.41.3.50 listTagsWithNoteLocalUids()	128
4.41.3.51 localStorageRequiresUpgrade()	129
4.41.3.52 localStorageVersion()	129
4.41.3.53 notebookCount()	130
4.41.3.54 noteCount()	130
4.41.3.55 noteCountPerNotebook()	130
4.41.3.56 noteCountPerNotebooksAndTags()	131
4.41.3.57 noteCountPerTag()	131



4.41.3.58 noteCountsPerAllTags()	132
4.41.3.59 requiredLocalStoragePatches()	132
4.41.3.60 savedSearchCount()	133
4.41.3.61 switchUser()	133
4.41.3.62 tagCount()	133
4.41.3.63 updateEnResource()	134
4.41.3.64 updateLinkedNotebook()	134
4.41.3.65 updateNote()	135
4.41.3.66 updateNotebook()	135
4.41.3.67 updateSavedSearch()	136
4.41.3.68 updateTag()	136
4.41.3.69 updateUser()	137
4.41.3.70 upgradeProgress	137
4.41.3.71 userCount()	138
4.42 quantier::LocalStorageManagerAsync Class Reference	138
4.43 quantier::LoggerInitializationException Class Reference	145
4.44 quantier::LRUCache< Key, Value, Allocator > Class Template Reference	146
4.45 quantier::Note Class Reference	147
4.46 quantier::Notebook Class Reference	150
4.47 quantier::ENMLConverter::NoteContentToHtmlExtraData Struct Reference	154
4.48 quantier::NoteEditor Class Reference	154
4.48.1 Detailed Description	157
4.48.2 Member Function Documentation	157
4.48.2.1 backend()	158
4.48.2.2 clear()	158
4.48.2.3 convertToNote	158
4.48.2.4 currentNoteLocalUid()	158
4.48.2.5 defaultFont()	158
4.48.2.6 defaultPalette()	158
4.48.2.7 idleTime()	159
4.48.2.8 inAppNoteLinkPasteRequested	159
4.48.2.9 initialize()	159
4.48.2.10 isEditorPageModified()	160
4.48.2.11 isModified()	160
4.48.2.12 isNoteLoaded()	160
4.48.2.13 saveNoteToLocalStorage	160
4.48.2.14 setAccount()	160
4.48.2.15 setBackend()	160
4.48.2.16 setCurrentNoteLocalUid()	160
4.48.2.17 setDefaultFont	161
4.48.2.18 setDefaultPalette	161
4.48.2.19 setFocus()	161

4.48.2.20 setInitialPageHtml()	162
4.48.2.21 setNoteDeletedPageHtml()	162
4.48.2.22 setNoteLoadingPageHtml()	162
4.48.2.23 setNoteNotFoundPageHtml()	162
4.48.2.24 setNoteTitle	162
4.48.2.25 setTagIds	162
4.48.2.26 setUndoStack()	163
4.48.2.27 undoStack()	163
4.49 quantier::NoteEditorInitializationException Class Reference	163
4.50 quantier::NoteEditorPluginInitializationException Class Reference	164
4.51 quantier::NoteSearchQuery Class Reference	165
4.51.1 Member Function Documentation	168
4.51.1.1 notebookModifier()	168
4.51.1.2 queryString()	168
4.52 quantier::NullPtrException Class Reference	168
4.53 quantier::ResourceRecognitionIndexItem::ObjectItem Struct Reference	169
4.54 quantier::Printable Class Reference	170
4.54.1 Detailed Description	171
4.55 quantier::QuantierApplication Class Reference	171
4.56 quantier::QuantierUndoCommand Class Reference	172
4.56.1 Detailed Description	173
4.57 quantier::Resource Class Reference	173
4.58 quantier::ResourceRecognitionIndexItem Class Reference	176
4.59 quantier::ResourceRecognitionIndices Class Reference	178
4.60 quantier::SavedSearch Class Reference	179
4.61 quantier::ResourceRecognitionIndexItem::ShapeItem Struct Reference	181
4.62 quantier::SharedNote Class Reference	182
4.63 quantier::SharedNotebook Class Reference	184
4.64 quantier::ShortcutManager Class Reference	186
4.64.1 Member Function Documentation	187
4.64.1.1 defaultShortcut() [1/2]	188
4.64.1.2 defaultShortcut() [2/2]	188
4.64.1.3 shortcut() [1/2]	188
4.64.1.4 shortcut() [2/2]	188
4.64.1.5 userShortcut() [1/2]	189
4.64.1.6 userShortcut() [2/2]	189
4.65 quantier::ENMLConverter::SkipHtmlElementRule Class Reference	189
4.65.1 Detailed Description	190
4.66 quantier::SpellChecker Class Reference	191
4.67 quantier::StringUtils::StringFilterPredicate Struct Reference	192
4.68 quantier::StringUtils Class Reference	192
4.69 quantier::SynchronizationManager Class Reference	192

4.69.1 Detailed Description	194
4.69.2 Constructor & Destructor Documentation	194
4.69.2.1 SynchronizationManager()	194
4.69.3 Member Function Documentation	195
4.69.3.1 active()	195
4.69.3.2 authenticate	195
4.69.3.3 authenticateCurrentAccount	195
4.69.3.4 authenticationFinished	195
4.69.3.5 authenticationRevoked	196
4.69.3.6 detectedConflictDuringLocalChangesSending	196
4.69.3.7 downloadNoteThumbnailsOption()	196
4.69.3.8 failed	197
4.69.3.9 finished	197
4.69.3.10 linkedNotebooksNotesDownloadProgress	197
4.69.3.11 linkedNotebooksResourcesDownloadProgress	197
4.69.3.12 linkedNotebooksSyncChunksDownloaded	198
4.69.3.13 linkedNotebookSyncChunksDownloadProgress	198
4.69.3.14 notesDownloadProgress	198
4.69.3.15 preparedDirtyObjectsForSending	199
4.69.3.16 preparedLinkedNotebooksDirtyObjectsForSending	199
4.69.3.17 rateLimitExceeded	199
4.69.3.18 remoteToLocalSyncDone	199
4.69.3.19 remoteToLocalSyncStopped	200
4.69.3.20 resourcesDownloadProgress	200
4.69.3.21 revokeAuthentication	200
4.69.3.22 sendLocalChangesStopped	200
4.69.3.23 setAccount	201
4.69.3.24 setAccountDone	201
4.69.3.25 setDownloadInkNoteImages	201
4.69.3.26 setDownloadInkNoteImagesDone	201
4.69.3.27 setDownloadNoteThumbnails	201
4.69.3.28 setDownloadNoteThumbnailsDone	202
4.69.3.29 setInkNoteImagesStoragePath	202
4.69.3.30 setInkNoteImagesStoragePathDone	202
4.69.3.31 started	202
4.69.3.32 stop	202
4.69.3.33 stopped	202
4.69.3.34 syncChunksDownloaded	203
4.69.3.35 syncChunksDownloadProgress	203
4.69.3.36 synchronize	203
4.69.3.37 willRepeatRemoteToLocalSyncAfterSendingChanges	203
4.70 quotienter::SysInfo Class Reference	203

4.71 <a href="#">quentier::Tag Class Reference</a> . . . . .	204
4.72 <a href="#">quentier::ResourceRecognitionIndexItem::TextItem Struct Reference</a> . . . . .	206
4.73 <a href="#">quentier::UidGenerator Class Reference</a> . . . . .	206
4.74 <a href="#">quentier::User Class Reference</a> . . . . .	207
<b><a href="#">Index</a></b>	<b>211</b>

# Chapter 1

## libquentier

Set of Qt/C++ APIs for feature rich desktop clients for Evernote service

### 1.1 What's this

This library presents a set of Qt/C++ APIs useful for applications working as feature rich desktop clients for Evernote service. The most important and useful components of the library are the following:

- Local storage - persistence of data downloaded from Evernote service in a local SQLite database
- Synchronization - the logics of exchanging new and/or modified data with Evernote service
- Note editor - the UI component capable for notes displaying and editing

The library is based on the lower level functionality provided by `QEverCloud` library. It also serves as the functional core of `Quentier` application.

**1.1.1 WARNING: libquentier is in alpha state right now, neither API nor ABI can be considered stable yet!**

### 1.2 How to build/install

Please see the building/installation guide.

### 1.3 How to contribute

Please see the contribution guide for detailed info.



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

quentier::ApplicationSettings::ArrayCloser . . . . .	22
quentier::ResourceRecognitionIndexItem::BarcodeItem . . . . .	24
quentier::DateTimePrint . . . . .	29
quentier::DecryptedTextManager . . . . .	29
quentier::ENMLConverter . . . . .	36
std::exception	
quentier::IQuentierException . . . . .	82
quentier::ApplicationSettingsInitializationException . . . . .	21
quentier::DatabaseLockFailedException . . . . .	25
quentier::DatabaseLockedException . . . . .	24
quentier::DatabaseOpeningException . . . . .	26
quentier::DatabaseRequestException . . . . .	27
quentier::EmptyDataElementException . . . . .	34
quentier::LocalStorageCacheManagerException . . . . .	94
quentier::LoggerInitializationException . . . . .	145
quentier::NoteEditorInitializationException . . . . .	163
quentier::NoteEditorPluginInitializationException . . . . .	164
quentier::NullPtrException . . . . .	168
quentier::ApplicationSettings::GroupCloser . . . . .	48
quentier::HTMLCleaner . . . . .	49
quentier::ILocalStorageDataElement . . . . .	60
quentier::INoteStoreDataElement . . . . .	81
quentier::IFavoritableDataElement . . . . .	50
quentier::Note . . . . .	147
quentier::Notebook . . . . .	150
quentier::SavedSearch . . . . .	179
quentier::Tag . . . . .	204
quentier::LinkedNotebook . . . . .	90
quentier::Resource . . . . .	173
quentier::INoteEditorBackend . . . . .	66
quentier::IUserStore . . . . .	87
quentier::LRUCache< Key, Value, Allocator > . . . . .	146
quentier::ENMLConverter::NoteContentToHtmlExtraData . . . . .	154
quentier::ResourceRecognitionIndexItem::ObjectItem . . . . .	169
quentier::Printable . . . . .	170

quentier::Account	9
quentier::ApplicationSettings	13
quentier::ENMLConverter::SkipHtmlElementRule	189
quentier::ErrorString	39
quentier::ILocalStorageCacheExpiryChecker	57
quentier::DefaultLocalStorageCacheExpiryChecker	30
quentier::INoteStoreDataElement	81
quentier::IQuentierException	82
quentier::ISyncStateStorage::ISyncState	84
quentier::LocalStorageCacheManager	92
quentier::NoteSearchQuery	165
quentier::ResourceRecognitionIndexItem	176
quentier::ResourceRecognitionIndices	178
quentier::SharedNote	182
quentier::SharedNotebook	184
quentier::User	207
QApplication	
quentier::QuentierApplication	171
QEventLoop	
quentier::EventLoopWithExitStatus	41
QObject	
quentier::EncryptionManager	35
quentier::FileCopier	42
quentier::FileIOProcessorAsync	44
quentier::FileSystemWatcher	47
quentier::IAuthenticationManager	49
quentier::AuthenticationManager	23
quentier::IKeychainService	51
quentier::ILocalStoragePatch	61
quentier::INoteStore	69
quentier::ISyncStateStorage	85
quentier::LocalStorageManager	95
quentier::LocalStorageManagerAsync	138
quentier::QuentierUndoCommand	172
quentier::ShortcutManager	186
quentier::SpellChecker	191
quentier::SynchronizationManager	192
QSettings	
quentier::ApplicationSettings	13
QStack	
quentier::LimitedStack< T >	89
QUndoCommand	
quentier::QuentierUndoCommand	172
QWidget	
quentier::NoteEditor	154
quentier::ResourceRecognitionIndexItem::ShapeItem	181
quentier::StringUtils::StringFilterPredicate	192
quentier::StringUtils	192
quentier::SysInfo	203
quentier::ResourceRecognitionIndexItem::TextItem	206
quentier::UidGenerator	206



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">quentier::Account</a>	Encapsulates some details about the account: its name, whether it is local or synchronized to Evernote and for the latter case - some additional details like upload limit etc . . . . .	9
<a href="#">quentier::ApplicationSettings</a>	Enhances the functionality of QSettings, in particular it simplifies the way of working with either application-wide or account-specific settings . . . . .	13
<a href="#">quentier::ApplicationSettingsInitializationException</a>	The <a href="#">ApplicationSettingsInitializationException</a> can be thrown from methods of <a href="#">ApplicationSettings</a> class if it's unable to locate the file with persistent settings . . . . .	21
<a href="#">quentier::ApplicationSettings::ArrayCloser</a>		22
<a href="#">quentier::AuthenticationManager</a>	Libquentier's default implementation of <a href="#">IAuthenticationManager</a> interface; internally uses QEverCloud's OAuth widget . . . . .	23
<a href="#">quentier::ResourceRecognitionIndexItem::BarcodeItem</a>		24
<a href="#">quentier::DatabaseLockedException</a>		24
<a href="#">quentier::DatabaseLockFailedException</a>		25
<a href="#">quentier::DatabaseOpeningException</a>		26
<a href="#">quentier::DatabaseRequestException</a>	The <a href="#">DatabaseRequestException</a> is thrown when the local storage database encounters some internal error during the attempt to serve a request . . . . .	27
<a href="#">quentier::DateTimePrint</a>	Simply wraps the enum containing datetime printing options . . . . .	29
<a href="#">quentier::DecryptedTextManager</a>		29
<a href="#">quentier::DefaultLocalStorageCacheExpiryChecker</a>		30
<a href="#">quentier::EmptyDataElementException</a>		34
<a href="#">quentier::EncryptionManager</a>	Both synchronous methods to encrypt or decrypt given text with password, cipher and key length and their signal-slot based potentially asynchronous counterparts . . . . .	35
<a href="#">quentier::ENMLConverter</a>	Encapsulates a set of methods and helper data structures for performing the conversions between ENML and other note content formats, namely HTML . . . . .	36
<a href="#">quentier::ErrorString</a>	Encapsulates two (or more) strings which are meant to contain translatable (base) and non-translatable (details) parts of the error description . . . . .	39
<a href="#">quentier::EventLoopWithExitStatus</a>		41

<a href="#">quentier::FileCopier</a>	42
<a href="#">quentier::FileIOProcessorAsync</a>	
Wrapper under simple file IO operations, it is meant to be used for simple asynchronous IO	44
<a href="#">quentier::FileSystemWatcher</a>	47
<a href="#">quentier::ApplicationSettings::GroupCloser</a>	48
<a href="#">quentier::HTMLCleaner</a>	49
<a href="#">quentier::IAuthenticationManager</a>	49
<a href="#">quentier::IFavoritableDataElement</a>	50
<a href="#">quentier::IKeychainService</a>	
The <a href="#">IKeychainService</a> interface provides methods intended to start potentially asynchronous interaction with the keychain and signals intended to notify listeners about the completion of asynchronous interactions	51
<a href="#">quentier::ILocalStorageCacheExpiryChecker</a>	
Interface for cache expiry checker used by <a href="#">LocalStorageCacheManager</a> to see whether particular caches (of notes, notebooks, tags, linked notebooks and/or saved searches) need to be shrunk	57
<a href="#">quentier::ILocalStorageDataElement</a>	60
<a href="#">quentier::ILocalStoragePatch</a>	
Interface for patches of local storage. Each such patch somehow changes the layout of local storage persistence so that only compliant & corresponding versions of libquentier can be used to work with it	61
<a href="#">quentier::INoteEditorBackend</a>	66
<a href="#">quentier::INoteStore</a>	
<a href="#">INoteStore</a> is the interface which provides methods required for the implementation of NoteStore part of Evernote EDAM sync protocol	69
<a href="#">quentier::INoteStoreDataElement</a>	81
<a href="#">quentier::IQuentierException</a>	
Interface for exceptions specific to libquentier and applications based on it	82
<a href="#">quentier::ISyncStateStorage::ISyncState</a>	
The <a href="#">ISyncState</a> interface provides accessory methods to determine the sync state for the account	84
<a href="#">quentier::ISyncStateStorage</a>	
The <a href="#">ISyncStateStorage</a> interface represents the interface of a class which stores sync state for given accounts persistently and provides access to previously stores sync states	85
<a href="#">quentier::IUserStore</a>	
<a href="#">IUserStore</a> is the interface which provides methods required for the implementation of UserStore part of Evernote EDAM sync protocol	87
<a href="#">quentier::LimitedStack&lt; T &gt;</a>	
The <a href="#">LimitedStack</a> template class implements a stack which may have a limitation for its size; when the size becomes too much according to the limit, the bottom element of the stack gets erased from it. Only limits greater than zero are considered	89
<a href="#">quentier::LinkedNotebook</a>	90
<a href="#">quentier::LocalStorageCacheManager</a>	92
<a href="#">quentier::LocalStorageCacheManagerException</a>	94
<a href="#">quentier::LocalStorageManager</a>	95
<a href="#">quentier::LocalStorageManagerAsync</a>	138
<a href="#">quentier::LoggerInitializationException</a>	145
<a href="#">quentier::LRUCache&lt; Key, Value, Allocator &gt;</a>	146
<a href="#">quentier::Note</a>	147
<a href="#">quentier::Notebook</a>	150
<a href="#">quentier::ENMLConverter::NoteContentToHtmlExtraData</a>	154
<a href="#">quentier::NoteEditor</a>	
Widget encapsulating all the functionality necessary for showing and editing notes	154
<a href="#">quentier::NoteEditorInitializationException</a>	163
<a href="#">quentier::NoteEditorPluginInitializationException</a>	164
<a href="#">quentier::NoteSearchQuery</a>	165
<a href="#">quentier::NullPtrException</a>	168
<a href="#">quentier::ResourceRecognitionIndexItem::ObjectItem</a>	169

<a href="#">quentier::Printable</a>	
Interface for Quentier's internal classes which should be able to write themselves into QText↔ Stream and/or convert to QString . . . . .	170
<a href="#">quentier::QuentierApplication</a> . . . . .	171
<a href="#">quentier::QuentierUndoCommand</a>	
Has the sole purpose of working around one quirky aspect of Qt's undo/redo framework: when you push QUndoCommand to QUndoStack, it calls "redo" method of that command. This class offers subclasses to implement their own methods for actual "undo" and "redo" commands while ignoring the attempts to "redo" anything if there were no previous "undo" call prior to that . . .	172
<a href="#">quentier::Resource</a> . . . . .	173
<a href="#">quentier::ResourceRecognitionIndexItem</a> . . . . .	176
<a href="#">quentier::ResourceRecognitionIndices</a> . . . . .	178
<a href="#">quentier::SavedSearch</a> . . . . .	179
<a href="#">quentier::ResourceRecognitionIndexItem::ShapelItem</a> . . . . .	181
<a href="#">quentier::SharedNote</a> . . . . .	182
<a href="#">quentier::SharedNotebook</a> . . . . .	184
<a href="#">quentier::ShortcutManager</a> . . . . .	186
<a href="#">quentier::ENMLConverter::SkipHtmlElementRule</a>	
Describes the set of rules for HTML -> ENML conversion about the HTML elements that should not be actually converted to ENML due to their nature of being "helper" elements for the display or functioning of something within the note editor's page. The HTML -> ENML conversion would ignore tags and attributes forbidden by ENML even without these rules conditionally preserving or skipping the contents and nested elements of skipped elements . . . . .	189
<a href="#">quentier::SpellChecker</a> . . . . .	191
<a href="#">quentier::StringUtils::StringFilterPredicate</a> . . . . .	192
<a href="#">quentier::StringUtils</a> . . . . .	192
<a href="#">quentier::SynchronizationManager</a>	
Encapsulates methods and signals & slots required to perform the full or partial synchronization of data with remote Evernote servers. The class also deals with authentication with Evernote service through OAuth . . . . .	192
<a href="#">quentier::SysInfo</a> . . . . .	203
<a href="#">quentier::Tag</a> . . . . .	204
<a href="#">quentier::ResourceRecognitionIndexItem::TextItem</a> . . . . .	206
<a href="#">quentier::UidGenerator</a> . . . . .	206
<a href="#">quentier::User</a> . . . . .	207



## Chapter 4

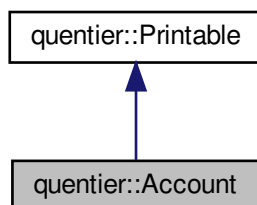
# Class Documentation

### 4.1 `quentier::Account` Class Reference

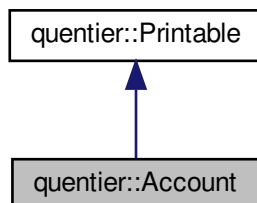
The [Account](#) class encapsulates some details about the account: its name, whether it is local or synchronized to Evernote and for the latter case - some additional details like upload limit etc.

```
#include <Account.h>
```

Inheritance diagram for `quentier::Account`:



Collaboration diagram for `quentier::Account`:



## Public Types

- enum class **Type** { **Local** = 0 , **Evernote** }
- enum class **EvernoteAccountType** { **Free** = 0 , **Plus** , **Premium** , **Business** }

## Public Member Functions

- **Account** (QString **name**, const Type **type**, const qevercloud::UserID **userId**=-1, const EvernoteAccountType **evernoteAccountType**=EvernoteAccountType::Free, QString **evernoteHost**={}, QString **shardId**={})
- **Account** (const **Account** &**other**)
- **Account** & **operator=** (const **Account** &**other**)
- bool **operator==** (const **Account** &**other**) const
- bool **operator!=** (const **Account** &**other**) const
- bool **isEmpty** () const
- QString **name** () const
- void **setName** (QString **name**)  
*setName sets the username to the account*
- QString **displayName** () const
- void **setDisplayName** (QString **displayName**)
- Type **type** () const
- qevercloud::UserID **id** () const
- EvernoteAccountType **evernoteAccountType** () const
- QString **evernoteHost** () const
- QString **shardId** () const
- void **setEvernoteAccountType** (const EvernoteAccountType **evernoteAccountType**)
- void **setEvernoteHost** (QString **evernoteHost**)
- void **setShardId** (QString **shardId**)
- qint32 **mailLimitDaily** () const
- qint64 **noteSizeMax** () const
- qint64 **resourceSizeMax** () const
- qint32 **linkedNotebookMax** () const
- qint32 **noteCountMax** () const
- qint32 **notebookCountMax** () const
- qint32 **tagCountMax** () const
- qint32 **noteTagCountMax** () const
- qint32 **savedSearchCountMax** () const
- qint32 **noteResourceCountMax** () const
- void **setEvernoteAccountLimits** (const qevercloud::AccountLimits &**limits**)
- virtual QTextStream & **print** (QTextStream &**strm**) const override

## Friends

- QUENTIER\_EXPORT QTextStream & **operator<<** (QTextStream &**strm**, const Type **type**)
- QUENTIER\_EXPORT QDebug & **operator<<** (QDebug &**dbg**, const Type **type**)
- QUENTIER\_EXPORT QTextStream & **operator<<** (QTextStream &**strm**, const EvernoteAccountType **type**)
- QUENTIER\_EXPORT QDebug & **operator<<** (QDebug &**dbg**, const EvernoteAccountType **type**)

## Additional Inherited Members

### 4.1.1 Detailed Description

The **Account** class encapsulates some details about the account: its name, whether it is local or synchronized to Evernote and for the latter case - some additional details like upload limit etc.

## 4.1.2 Member Function Documentation

### 4.1.2.1 displayName()

```
QString quentier::Account::displayName ( ) const
```

#### Returns

[Printable](#) user's name that is not used to uniquely identify the account, so this name may repeat across different local and Evernote accounts

### 4.1.2.2 evernoteAccountType()

```
EvernoteAccountType quentier::Account::evernoteAccountType ( ) const
```

#### Returns

The type of the Evernote account; if applied to free account, returns "Free"

### 4.1.2.3 evernoteHost()

```
QString quentier::Account::evernoteHost ( ) const
```

#### Returns

The Evernote server host with which the account is associated

### 4.1.2.4 id()

```
qevercloud::UserID quentier::Account::id ( ) const
```

#### Returns

[User](#) id for Evernote accounts, -1 for local accounts (as the concept of user id is not defined for local accounts)

#### 4.1.2.5 isEmpty()

```
bool quentier::Account::isEmpty ( ) const
```

##### Returns

True if either the account is local but the name is empty or if the account is Evernote but user id is negative; in all other cases return false

#### 4.1.2.6 name()

```
QString quentier::Account::name ( ) const
```

##### Returns

Username for either local or Evernote account

#### 4.1.2.7 setDisplayName()

```
void quentier::Account::setDisplayName (
    QString displayName )
```

Set the printable name of the account

#### 4.1.2.8 shardId()

```
QString quentier::Account::shardId ( ) const
```

##### Returns

Shard id for Evernote accounts, empty string for local accounts (as the concept of shard id is not defined for local accounts)

#### 4.1.2.9 type()

```
Type quentier::Account::type ( ) const
```

##### Returns

The type of the account: either local or Evernote

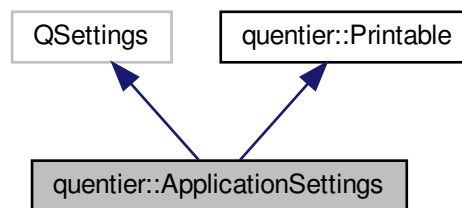


## 4.2 `quentier::ApplicationSettings` Class Reference

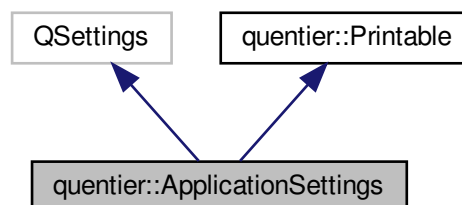
The [ApplicationSettings](#) class enhances the functionality of `QSettings`, in particular it simplifies the way of working with either application-wide or account-specific settings.

```
#include <ApplicationSettings.h>
```

Inheritance diagram for `quentier::ApplicationSettings`:



Collaboration diagram for `quentier::ApplicationSettings`:



### Classes

- struct [ArrayCloser](#)
- struct [GroupCloser](#)

### Public Member Functions

- [ApplicationSettings](#) (const `QString` &settingsName={})
- [ApplicationSettings](#) (const [Account](#) &account, const `QString` &settingsName={})
- [ApplicationSettings](#) (const [Account](#) &account, const `char` \*settingsName, const `int` settingsNameSize=-1)
- virtual [~ApplicationSettings](#) () override
- void [beginGroup](#) (const `QString` &prefix)

- void [beginGroup](#) (const char \*prefix, const int size=-1)
- int [beginReadArray](#) (const QString &prefix)
- int [beginReadArray](#) (const char \*prefix, const int size=-1)
- void [beginWriteArray](#) (const QString &prefix, const int arraySize=-1)
- void [beginWriteArray](#) (const char \*prefix, const int arraySize=-1, const int prefixSize=-1)
- bool [contains](#) (const QString &key) const
- bool [contains](#) (const char \*key, const int size=-1) const
- void [remove](#) (const QString &key)
- void [remove](#) (const char \*key, const int size=-1)
- void [setValue](#) (const QString &key, const QVariant &value)
- void [setValue](#) (const char \*key, const QVariant &value, const int keySize=-1)
- QVariant [value](#) (const QString &key, const QVariant &defaultValue={}) const
- QVariant [value](#) (const char \*key, const QVariant &defaultValue={}, const int keySize=-1) const
- virtual QTextStream & [print](#) (QTextStream &strm) const override

## Additional Inherited Members

### 4.2.1 Detailed Description

The [ApplicationSettings](#) class enhances the functionality of QSettings, in particular it simplifies the way of working with either application-wide or account-specific settings.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 ApplicationSettings() [1/3]

```
quentier::ApplicationSettings::ApplicationSettings (
    const QString & settingsName = {} )
```

Constructor for application settings not being account-specific

##### Parameters

<i>settingsName</i>	If not empty, the created application settings would manage the settings stored in a file with a specific name within the common settings storage; otherwise they would be stored in the default settings file for the account
---------------------	--

#### 4.2.2.2 ApplicationSettings() [2/3]

```
quentier::ApplicationSettings::ApplicationSettings (
    const Account & account,
    const QString & settingsName = {} )
```

Constructor for application settings specific to the account

## Parameters

<i>account</i>	The account for which the settings are to be stored or read
<i>settingsName</i>	If not empty, the created application settings would manage the settings stored in a file with a specific name within the account's settings storage; otherwise they would be stored in the default settings file for the account

## 4.2.2.3 ApplicationSettings() [3/3]

```
quentier::ApplicationSettings::ApplicationSettings (
    const Account & account,
    const char * settingsName,
    const int settingsNameSize = -1 )
```

Constructor for application settings specific to the account

## Parameters

<i>account</i>	The account for which the settings are to be stored or read
<i>settingsName</i>	If not nullptr, the created application settings would manage the settings stored in a file with a specific name within the account's settings storage; otherwise they would be stored in the default settings file for the account. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>settingsNameSize</i>	Size of the settingsName string. If negative (the default), the settingsName size is taken to be strlen(settingsName)

## 4.2.2.4 ~ApplicationSettings()

```
virtual quentier::ApplicationSettings::~~ApplicationSettings ( ) [override], [virtual]
```

Destructor

## 4.2.3 Member Function Documentation

## 4.2.3.1 beginGroup() [1/2]

```
void quentier::ApplicationSettings::beginGroup (
    const char * prefix,
    const int size = -1 )
```

Appends prefix to the current group. Overload of beginGroup accepting const char \* and optionally the size of the string

## Parameters

<i>prefix</i>	String containing the prefix name. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>size</i>	Size of the prefix string. If negative (the default), the prefix size is taken to be strlen(prefix).

**4.2.3.2 beginGroup() [2/2]**

```
void quantier::ApplicationSettings::beginGroup (
    const QString & prefix )
```

Appends prefix to the current group. The call is redirected to QSettings::beginGroup. It is required in this class only to workaround hiding QSettings method due to overloads

## Parameters

<i>prefix</i>	String containing the prefix name
---------------	-----------------------------------

**4.2.3.3 beginReadArray() [1/2]**

```
int quantier::ApplicationSettings::beginReadArray (
    const char * prefix,
    const int size = -1 )
```

Adds prefix to the current group and starts reading from an array. Overload of beginReadArray accepting const char \* and optionally the size of the string

## Parameters

<i>prefix</i>	String containing the prefix name. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>size</i>	Size of the prefix string. If negative (the default), the prefix size is taken to be strlen(prefix)

**4.2.3.4 beginReadArray() [2/2]**

```
int quantier::ApplicationSettings::beginReadArray (
    const QString & prefix )
```

Adds prefix to the current group and starts reading from an array. The call is redirected to QSettings::beginReadArray. It is required in this class only to workaround hiding QSettings method due to overloads

## Parameters

<i>prefix</i>	String containing the prefix name
---------------	-----------------------------------

## Returns

The size of the array

**4.2.3.5 beginWriteArray() [1/2]**

```
void quentier::ApplicationSettings::beginWriteArray (
    const char * prefix,
    const int arraySize = -1,
    const int prefixSize = -1 )
```

Adds prefix to the current group and starts writing an array of size *arraySize*. Overload of *beginWriteArray* accepting *const char \** and optionally the size of the string

## Parameters

<i>prefix</i>	String containing the prefix name. Must be UTF-8 encoded as internally it is converted to <i>QString</i> via <i>QString::fromUtf8</i>
<i>arraySize</i>	Size of the array to be written. If negative (the default), it is automatically determined based on the indexes of the entries written.
<i>prefixSize</i>	Size of the prefix string. If negative (the default), the prefix size is taken to be <i>stren(prefix)</i>

**4.2.3.6 beginWriteArray() [2/2]**

```
void quentier::ApplicationSettings::beginWriteArray (
    const QString & prefix,
    const int arraySize = -1 )
```

Adds prefix to the current group and starts writing an array of size *arraySize*. The call is redirected to *QSettings::beginWriteArray*. It is required in this class only to workaround hiding *QSettings* method due to overloads

## Parameters

<i>prefix</i>	String containing the prefix name
<i>arraySize</i>	Size of the array to be written. If negative (the default), it is automatically determined based on the indexes of the entries written.

#### 4.2.3.7 contains() [1/2]

```
bool quentier::ApplicationSettings::contains (
    const char * key,
    const int size = -1 ) const
```

Overload of contains accepting const char \* and optionally the size of the string

##### Parameters

<i>key</i>	String containing the setting name. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>size</i>	Size of the key string. If negative (the default), the key size is taken to be strlen(key)

##### Returns

True if there exists a setting called key; false otherwise

#### 4.2.3.8 contains() [2/2]

```
bool quentier::ApplicationSettings::contains (
    const QString & key ) const
```

The call is redirected to QSettings::contains. It is required in this class only to workaround hiding QSettings method due to overloads

##### Parameters

<i>key</i>	The key being checked for presence
------------	------------------------------------

##### Returns

True if there exists a setting called key; false otherwise

#### 4.2.3.9 remove() [1/2]

```
void quentier::ApplicationSettings::remove (
    const char * key,
    const int size = -1 )
```

Removes the setting key and any sub-settings of key. Overload of remove accepting const char \* and optionally the size of the string

## Parameters

<i>key</i>	String containing the setting name. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>size</i>	Size of the key string. If negative (the default), the key size is taken to be strlen(key).

**4.2.3.10 remove()** [2/2]

```
void quentier::ApplicationSettings::remove (
    const QString & key )
```

Removes the setting key and any sub-settings of key. The call is redirected to QSettings::remove. It is required in this class only to workaround hiding QSettings method due to overloads

## Parameters

<i>key</i>	String containing the setting name
------------	------------------------------------

**4.2.3.11 setValue()** [1/2]

```
void quentier::ApplicationSettings::setValue (
    const char * key,
    const QVariant & value,
    const int keySize = -1 )
```

Sets the value of setting. Overload of setValue accepting const char \* and optionally the size of the string

## Parameters

<i>key</i>	String containing the setting name. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>value</i>	Value for setting key
<i>keySize</i>	Size of the key string. If negative (the default), the key size is taken to be strlen(key).

**4.2.3.12 setValue()** [2/2]

```
void quentier::ApplicationSettings::setValue (
    const QString & key,
    const QVariant & value )
```

Sets the value of setting. The call is redirected to QSettings::setValue. It is required in this class only to workaround hiding QSettings method due to overloads

## Parameters

<i>key</i>	String containing the setting name
<i>value</i>	Value for setting key

**4.2.3.13 value()** [1/2]

```
QVariant quentier::ApplicationSettings::value (
    const char * key,
    const QVariant & defaultValue = {},
    const int keySize = -1 ) const
```

Fetches the value of setting. Overload of value accepting const char \* and optionally the size of the string

## Parameters

<i>key</i>	String containing the setting name. Must be UTF-8 encoded as internally it is converted to QString via QString::fromUtf8
<i>defaultValue</i>	Default value returned if the setting doesn't exist
<i>keySize</i>	Size of the key string. If negative (the default), the key size is taken to be strlen(key)

## Returns

The value for setting key. If the setting doesn't exist, returns defaultValue. If no default value is specified, a default QVariant is returned.

**4.2.3.14 value()** [2/2]

```
QVariant quentier::ApplicationSettings::value (
    const QString & key,
    const QVariant & defaultValue = {} ) const
```

Fetches the value of setting. The call is redirected to QSettings::value. It is required in this class only to workaround hiding QSettings method due to overloads

## Parameters

<i>key</i>	String containing the setting name
<i>defaultValue</i>	Default value returned if the setting doesn't exist

## Returns

The value for setting key. If the setting doesn't exist, returns defaultValue. If no default value is specified, a default QVariant is returned.

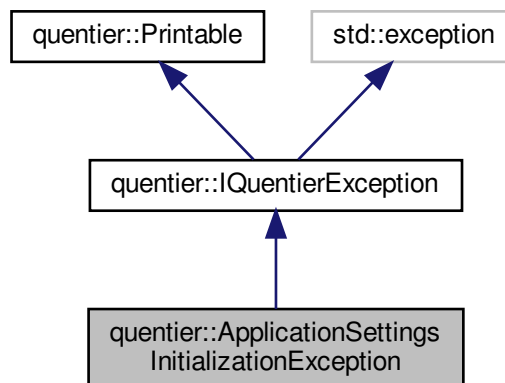


## 4.3 quantier::ApplicationSettingsInitializationException Class Reference

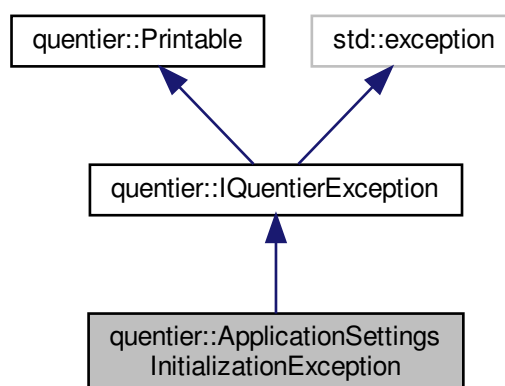
The [ApplicationSettingsInitializationException](#) can be thrown from methods of [ApplicationSettings](#) class if it's unable to locate the file with persistent settings.

```
#include <ApplicationSettingsInitializationException.h>
```

Inheritance diagram for quantier::ApplicationSettingsInitializationException:



Collaboration diagram for quantier::ApplicationSettingsInitializationException:



### Public Member Functions

- **ApplicationSettingsInitializationException** (const [ErrorString](#) &message)

## Protected Member Functions

- virtual const QString **exceptionDisplayName** () const override

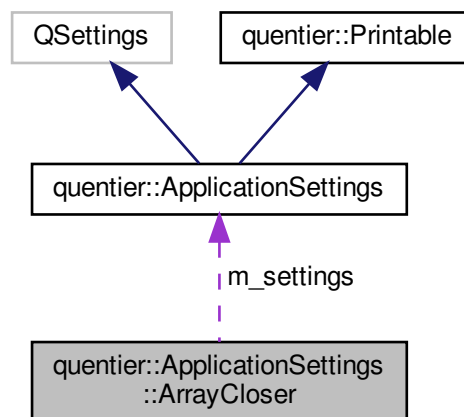
### 4.3.1 Detailed Description

The [ApplicationSettingsInitializationException](#) can be thrown from methods of [ApplicationSettings](#) class if it's unable to locate the file with persistent settings.

## 4.4 quantier::ApplicationSettings::ArrayCloser Struct Reference

```
#include <ApplicationSettings.h>
```

Collaboration diagram for quantier::ApplicationSettings::ArrayCloser:



## Public Member Functions

- **ArrayCloser** ([ApplicationSettings](#) &settings)

## Public Attributes

- [ApplicationSettings](#) & **m\_settings**

### 4.4.1 Detailed Description

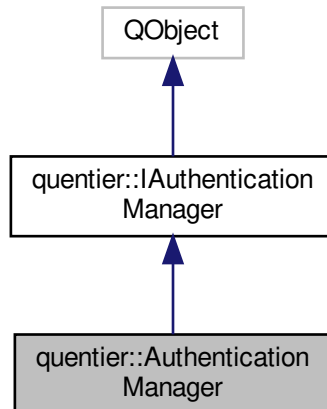
Helper struct for RAII style of ensuring the array once began would be closed even if exception is thrown after beginning the array

## 4.5 `quentier::AuthenticationManager` Class Reference

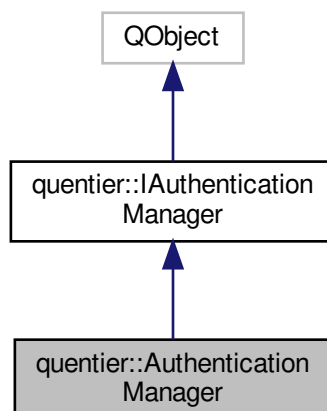
The `AuthenticationManager` class is libquentier's default implementation of `IAuthenticationManager` interface; internally uses QEverCloud's OAuth widget.

```
#include <AuthenticationManager.h>
```

Inheritance diagram for `quentier::AuthenticationManager`:



Collaboration diagram for `quentier::AuthenticationManager`:



### Public Slots

- virtual void **onAuthenticationRequest** () override

## Public Member Functions

- **AuthenticationManager** (const QString &consumerKey, const QString &consumerSecret, const QString &host, QObject \*parent=nullptr)

## Additional Inherited Members

### 4.5.1 Detailed Description

The [AuthenticationManager](#) class is libquentier's default implementation of [IAuthenticationManager](#) interface; internally uses QEverCloud's OAuth widget.

## 4.6 quentier::ResourceRecognitionIndexItem::BarcodeItem Struct Reference

### Public Member Functions

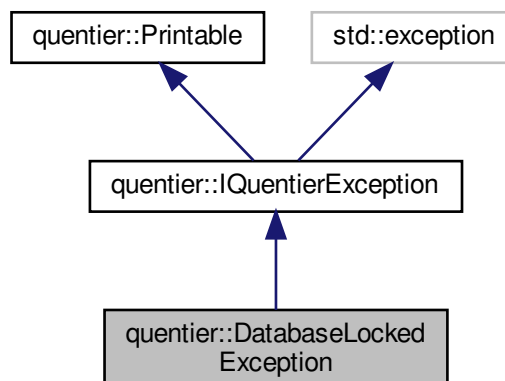
- bool **operator==** (const [BarcodeItem](#) &other) const

### Public Attributes

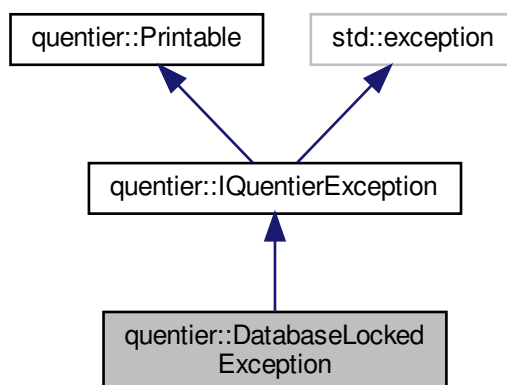
- QString **m\_barcode**
- int **m\_weight** = -1

## 4.7 quentier::DatabaseLockedException Class Reference

Inheritance diagram for quentier::DatabaseLockedException:



Collaboration diagram for `quentier::DatabaseLockedException`:



### Public Member Functions

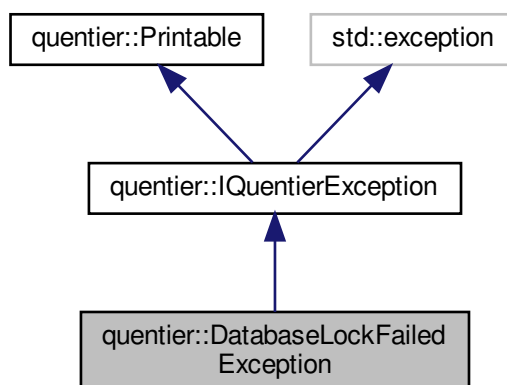
- **DatabaseLockedException** (const [ErrorString](#) &message)

### Protected Member Functions

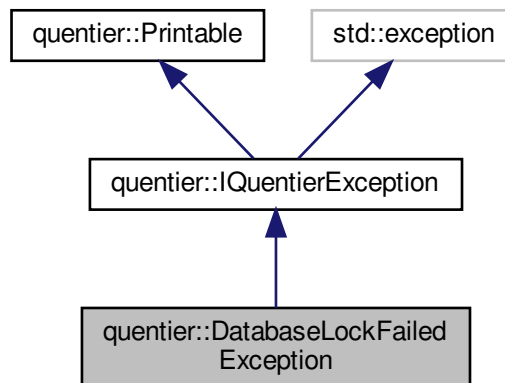
- virtual const QString **exceptionDisplayName** () const override

## 4.8 `quentier::DatabaseLockFailedException` Class Reference

Inheritance diagram for `quentier::DatabaseLockFailedException`:



Collaboration diagram for `quentier::DatabaseLockFailedException`:



### Public Member Functions

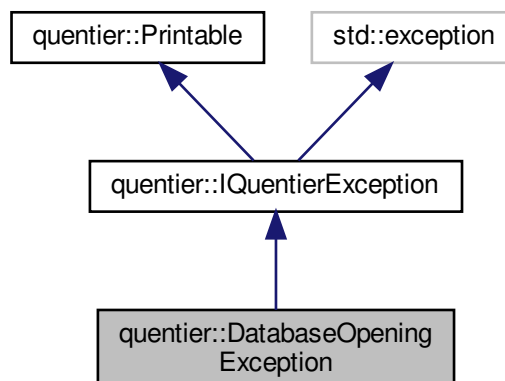
- **DatabaseLockFailedException** (const [ErrorString](#) &message)

### Protected Member Functions

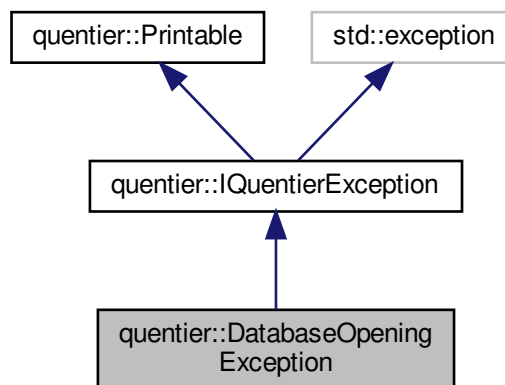
- virtual const QString **exceptionDisplayName** () const override

## 4.9 quentier::DatabaseOpeningException Class Reference

Inheritance diagram for `quentier::DatabaseOpeningException`:



Collaboration diagram for quantier::DatabaseOpeningException:



### Public Member Functions

- **DatabaseOpeningException** (const [ErrorString](#) &message)

### Protected Member Functions

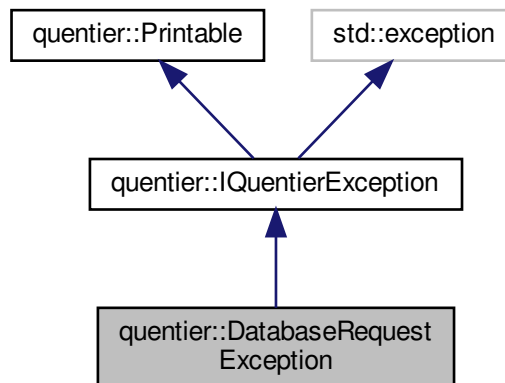
- virtual const QString **exceptionDisplayName** () const override

## 4.10 quantier::DatabaseRequestException Class Reference

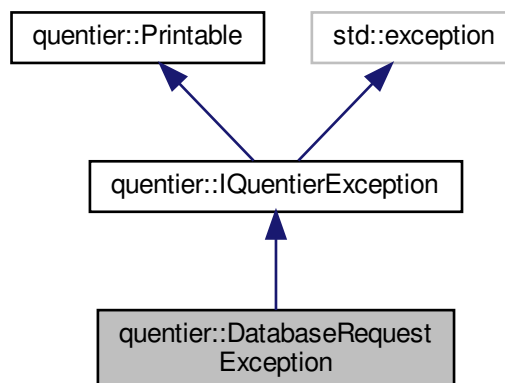
The [DatabaseRequestException](#) is thrown when the local storage database encounters some internal error during the attempt to serve a request.

```
#include <DatabaseRequestException.h>
```

Inheritance diagram for `quentier::DatabaseRequestException`:



Collaboration diagram for `quentier::DatabaseRequestException`:



## Public Member Functions

- `DatabaseRequestException` (const [ErrorString](#) &message)

## Protected Member Functions

- virtual const QString `exceptionDisplayName` () const override



### 4.10.1 Detailed Description

The [DatabaseRequestException](#) is thrown when the local storage database encounters some internal error during the attempt to serve a request.

## 4.11 quentier::DateTimePrint Class Reference

The [DateTimePrint](#) class simply wraps the enum containing datetime printing options.

```
#include <DateTime.h>
```

### Public Types

- enum [Option](#) { [IncludeNumericTimestamp](#) = 1 << 1 , [IncludeMilliseconds](#) = 1 << 2 , [IncludeTimezone](#) = 1 << 3 }

### 4.11.1 Detailed Description

The [DateTimePrint](#) class simply wraps the enum containing datetime printing options.

### 4.11.2 Member Enumeration Documentation

#### 4.11.2.1 Option

```
enum quentier::DateTimePrint::Option
```

Available printing options for datetime

Enumerator

<a href="#">IncludeNumericTimestamp</a>	Include the numeric representation of the timestamp into the printed string
<a href="#">IncludeMilliseconds</a>	Include milliseconds into the printed string
<a href="#">IncludeTimezone</a>	Include timezone into the printed string WARNING: currently this option has no effect on Windows platform, the timezone is not included anyway.

## 4.12 quentier::DecryptedTextManager Class Reference

### Public Member Functions

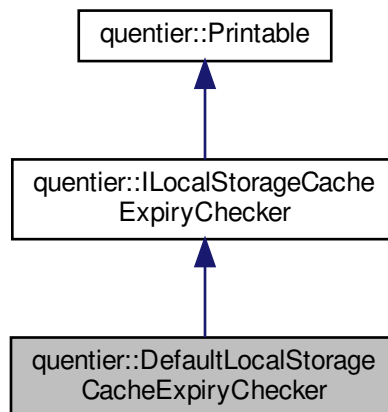
- void **addEntry** (const QString &hash, const QString &decryptedText, const bool rememberForSession, const QString &passphrase, const QString &cipher, const size\_t keyLength)

- void **removeEntry** (const QString &hash)
- void **clearNonRememberedForSessionEntries** ()
- bool **findDecryptedTextByEncryptedText** (const QString &encryptedText, QString &decryptedText, bool &rememberForSession) const
- bool **modifyDecryptedText** (const QString &originalEncryptedText, const QString &newDecryptedText, QString &newEncryptedText)

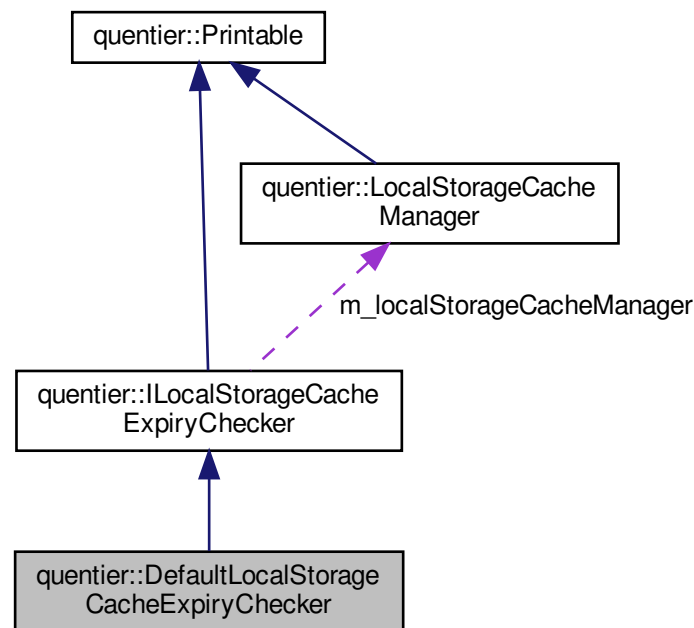
## 4.13 quantier::DefaultLocalStorageCacheExpiryChecker Class Reference

```
#include <DefaultLocalStorageCacheExpiryChecker.h>
```

Inheritance diagram for quantier::DefaultLocalStorageCacheExpiryChecker:



Collaboration diagram for quantier::DefaultLocalStorageCacheExpiryChecker:



## Public Member Functions

- **DefaultLocalStorageCacheExpiryChecker** (const [LocalStorageCacheManager](#) &cacheManager)
- virtual [DefaultLocalStorageCacheExpiryChecker](#) \* [clone](#) () const override
- virtual bool [checkNotes](#) () const override
- virtual bool [checkResources](#) () const override
- virtual bool [checkNotebooks](#) () const override
- virtual bool [checkTags](#) () const override
- virtual bool [checkLinkedNotebooks](#) () const override
- virtual bool [checkSavedSearches](#) () const override
- virtual QTextStream & [print](#) (QTextStream &strm) const override

*Print the internal information about the current [DefaultLocalStorageCacheExpiryChecker](#) instance to the text stream.*

## Additional Inherited Members

### 4.13.1 Detailed Description

brief The [DefaultLocalStorageCacheExpiryChecker](#) class is the implementation of [ILocalStorageCacheExpiryChecker](#) interface used by [LocalStorageCacheManager](#) by default, if no another implementation of [ILocalStorageCacheExpiryChecker](#) is set to be used by [LocalStorageCacheManager](#)

## 4.13.2 Member Function Documentation

### 4.13.2.1 `checkLinkedNotebooks()`

```
virtual bool quentier::DefaultLocalStorageCacheExpiryChecker::checkLinkedNotebooks ( ) const  
[override], [virtual]
```

#### Returns

False if the current number of cached linked notebooks is higher than a reasonable limit, true otherwise

Implements [quentier::ILocalStorageCacheExpiryChecker](#).

### 4.13.2.2 `checkNotebooks()`

```
virtual bool quentier::DefaultLocalStorageCacheExpiryChecker::checkNotebooks ( ) const [override],  
[virtual]
```

#### Returns

False if the current number of cached notebooks is higher than a reasonable limit, true otherwise

Implements [quentier::ILocalStorageCacheExpiryChecker](#).

### 4.13.2.3 `checkNotes()`

```
virtual bool quentier::DefaultLocalStorageCacheExpiryChecker::checkNotes ( ) const [override],  
[virtual]
```

#### Returns

False if the current number of cached notes is higher than a reasonable limit, true otherwise

Implements [quentier::ILocalStorageCacheExpiryChecker](#).

#### 4.13.2.4 `checkResources()`

```
virtual bool quentier::DefaultLocalStorageCacheExpiryChecker::checkResources ( ) const [override],  
[virtual]
```

##### Returns

False if the current number of cached resource is higher than a reasonable limit, true otherwise

Implements [quentier::ILocalStorageCacheExpiryChecker](#).

#### 4.13.2.5 `checkSavedSearches()`

```
virtual bool quentier::DefaultLocalStorageCacheExpiryChecker::checkSavedSearches ( ) const  
[override], [virtual]
```

##### Returns

False if the current number of cached saved searches is higher than a reasonable limit, true otherwise

Implements [quentier::ILocalStorageCacheExpiryChecker](#).

#### 4.13.2.6 `checkTags()`

```
virtual bool quentier::DefaultLocalStorageCacheExpiryChecker::checkTags ( ) const [override],  
[virtual]
```

##### Returns

False if the current number of cached tags is higher than a reasonable limit, true otherwise

Implements [quentier::ILocalStorageCacheExpiryChecker](#).

#### 4.13.2.7 `clone()`

```
virtual DefaultLocalStorageCacheExpiryChecker* quentier::DefaultLocalStorageCacheExpiryChecker::clone ( ) const [override], [virtual]
```

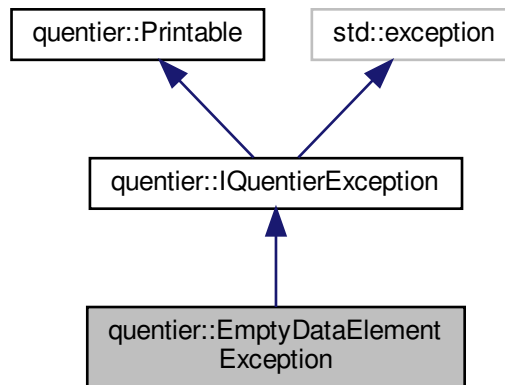
##### Returns

A pointer to the newly allocated copy of the current [DefaultLocalStorageCacheExpiryChecker](#)

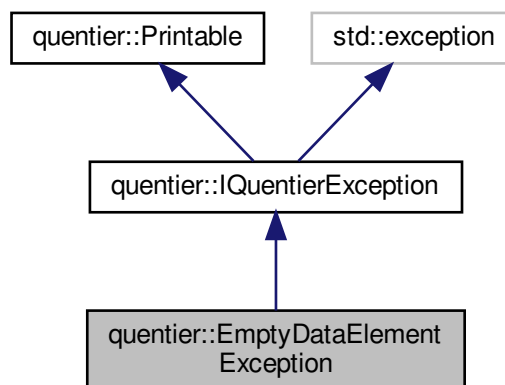
Implements [quentier::ILocalStorageCacheExpiryChecker](#).

## 4.14 quantier::EmptyDataElementException Class Reference

Inheritance diagram for quantier::EmptyDataElementException:



Collaboration diagram for quantier::EmptyDataElementException:



### Public Member Functions

- `EmptyDataElementException` (const [ErrorString](#) &message)

### Protected Member Functions

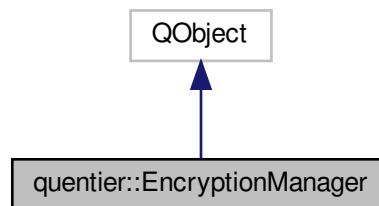
- virtual const QString `exceptionDisplayName` () const override

## 4.15 quantier::EncryptionManager Class Reference

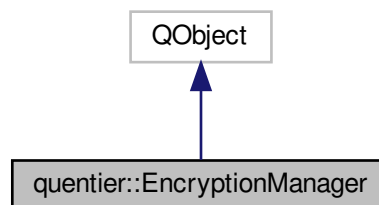
The [EncryptionManager](#) class provides both synchronous methods to encrypt or decrypt given text with password, cipher and key length and their signal-slot based potentially asynchronous counterparts.

```
#include <EncryptionManager.h>
```

Inheritance diagram for quantier::EncryptionManager:



Collaboration diagram for quantier::EncryptionManager:



### Public Slots

- void **onDecryptTextRequest** (QString encryptedText, QString passphrase, QString cipher, size\_t keyLength, QUuid requestId)
- void **onEncryptTextRequest** (QString textToEncrypt, QString passphrase, QString cipher, size\_t keyLength, QUuid requestId)

### Signals

- void **decryptedText** (QString text, bool success, [ErrorString](#) errorDescription, QUuid requestId)
- void **encryptedText** (QString encryptedText, bool success, [ErrorString](#) errorDescription, QUuid requestId)

## Public Member Functions

- **EncryptionManager** (QObject \*parent=nullptr)
- bool **decrypt** (const QString &encryptedText, const QString &passphrase, const QString &cipher, const size\_t keyLength, QString &decryptedText, [ErrorString](#) &errorDescription)
- bool **encrypt** (const QString &textToEncrypt, const QString &passphrase, QString &cipher, size\_t &keyLength, QString &encryptedText, [ErrorString](#) &errorDescription)

### 4.15.1 Detailed Description

The [EncryptionManager](#) class provides both synchronous methods to encrypt or decrypt given text with password, cipher and key length and their signal-slot based potentially asynchronous counterparts.

## 4.16 quotienter::ENMLConverter Class Reference

The [ENMLConverter](#) class encapsulates a set of methods and helper data structures for performing the conversions between ENML and other note content formats, namely HTML.

```
#include <ENMLConverter.h>
```

## Classes

- struct [NoteContentToHtmlExtraData](#)
- class [SkipHtmlElementRule](#)

*The [SkipHtmlElementRule](#) class describes the set of rules for HTML -> ENML conversion about the HTML elements that should not be actually converted to ENML due to their nature of being "helper" elements for the display or functioning of something within the note editor's page. The HTML -> ENML conversion would ignore tags and attributes forbidden by ENML even without these rules conditionally preserving or skipping the contents and nested elements of skipped elements.*

## Public Types

- enum class [EnexExportTags](#) { **Yes** = 0 , **No** }

*The [EnexExportTags](#) enum allows to specify whether export of note(s) to ENEX should include the names of note's tags.*

## Public Member Functions

- bool **htmlToNoteContent** (const QString &html, QString &noteContent, [DecryptedTextManager](#) &decryptedTextManager, [ErrorString](#) &errorDescription, const QVector< [SkipHtmlElementRule](#) > &skipRules=()) const
- bool **cleanupExternalHtml** (const QString &inputHtml, QString &cleanedUpHtml, [ErrorString](#) &errorDescription) const  
*cleanupExternalHtml method cleans up a piece of HTML coming from some external source: the cleanup includes the removal (or replacement with equivalents/alternatives) of any tags and attributes not supported by the ENML representation of note page's HTML*
- bool **htmlToQTextDocument** (const QString &html, QTextDocument &doc, [ErrorString](#) &errorDescription, const QVector< [SkipHtmlElementRule](#) > &skipRules=()) const



- bool **noteContentToHtml** (const QString &noteContent, QString &html, [ErrorString](#) &errorDescription, [DecryptedTextManager](#) &decryptedTextManager, [NoteContentToHtmlExtraData](#) &extraData) const
- bool **validateEnml** (const QString &enml, [ErrorString](#) &errorDescription) const
- bool **validateAndFixupEnml** (QString &enml, [ErrorString](#) &errorDescription) const
- bool **exportNotesToEnex** (const QVector< [Note](#) > &notes, const QHash< QString, QString > &tagNamesByTagLocalUids, const [EnexExportTags](#) exportTagsOption, QString &enex, [ErrorString](#) &errorDescription, const QString &version={}) const  
*exportNotesToEnex exports either a single note or a set of notes into ENEX format*
- bool **importEnex** (const QString &enex, QVector< [Note](#) > &notes, QHash< QString, QStringList > &tagNamesByNoteLocalUid, [ErrorString](#) &errorDescription) const  
*importEnex reads the content of input ENEX file and converts it into a set of notes and tag names.*

## Static Public Member Functions

- static bool **noteContentToPlainText** (const QString &noteContent, QString &plainText, [ErrorString](#) &error↵Message)
- static bool **noteContentToListOfWords** (const QString &noteContent, QStringList &listOfWords, [ErrorString](#) &errorMessage, QString \*plainText=nullptr)
- static QStringList **plainTextToListOfWords** (const QString &plainText)
- static QString **toDoCheckboxHtml** (const bool checked, const quint64 idNumber)
- static QString **encryptedTextHtml** (const QString &encryptedText, const QString &hint, const QString &ci-pher, const size\_t keyLength, const quint64 enCryptIndex)
- static QString **decryptedTextHtml** (const QString &decryptedText, const QString &encryptedText, const QString &hint, const QString &cipher, const size\_t keyLength, const quint64 enDecryptedIndex)
- static QString **resourceHtml** (const [Resource](#) &resource, [ErrorString](#) &errorDescription)
- static void **escapeString** (QString &string, const bool simplify=true)

### 4.16.1 Detailed Description

The [ENMLConverter](#) class encapsulates a set of methods and helper data structures for performing the conversions between ENML and other note content formats, namely HTML.

### 4.16.2 Member Function Documentation

#### 4.16.2.1 cleanupExternalHtml()

```
bool quantier::ENMLConverter::cleanupExternalHtml (
    const QString & inputHtml,
    QString & cleanedUpHtml,
    ErrorString & errorDescription ) const
```

cleanupExternalHtml method cleans up a piece of HTML coming from some external source: the cleanup includes the removal (or replacement with equivalents/alternatives) of any tags and attributes not supported by the ENML representation of note page's HTML

#### Parameters

<i>inputHtml</i>	- the input HTML to be cleaned up
<i>cleanedUpHtml</i>	- the result of the method's work
<i>errorDescription</i>	- the textual description of the error if conversion of input HTML into QTextDocument has failed

**Returns**

true in case of successful conversion, false otherwise

**4.16.2.2 exportNotesToEnex()**

```
bool quantier::ENMLConverter::exportNotesToEnex (
    const QVector< Note > & notes,
    const QHash< QString, QString > & tagNamesByTagLocalUids,
    const EnexExportTags exportTagsOption,
    QString & enex,
    ErrorString & errorDescription,
    const QString & version = {} ) const
```

exportNotesToEnex exports either a single note or a set of notes into ENEX format

**Parameters**

<i>notes</i>	The notes to be exported into the enex format. The connection of particular notes to tags is expected to follow from note's tag local uids. In other words, if some note has no tag local uids, its corresponding fragment of ENEX won't contain tag names associated with the note
<i>tagNamesByTagLocalUids</i>	Tag names for all tag local uids across all passed in notes. The lack of any tag name for any tag local uid is considered an error and the overall export attempt fails
<i>exportTagsOption</i>	Whether the export to ENEX should include the names of notes' tags
<i>enex</i>	The output of the method
<i>errorDescription</i>	The textual description of the error, if any
<i>version</i>	Optional "version" tag for the ENEX. If not set, the corresponding ENEX tag is set to empty value

**Returns**

True if the export completed successfully, false otherwise

**4.16.2.3 htmlToQTextDocument()**

```
bool quantier::ENMLConverter::htmlToQTextDocument (
    const QString & html,
    QTextDocument & doc,
    ErrorString & errorDescription,
    const QVector< SkipHtmlElementRule > & skipRules = {} ) const
```

Converts the passed in HTML into its simplified form acceptable by QTextDocument (see <http://doc.qt.io/qt-5/richtext-html-subset.html> for the list of elements supported by QTextDocument)

## Parameters

<i>html</i>	- the input HTML which needs to be converted to QTextDocument
<i>doc</i>	- QTextDocument filled with the result of the method's work
<i>errorDescription</i>	- the textual description of the error if conversion of input HTML into QTextDocument has failed
<i>skipRules</i>	- rules for skipping the particular elements

## Returns

true in case of successful conversion, false otherwise

## 4.16.2.4 importEnex()

```
bool quantier::ENMLConverter::importEnex (
    const QString & enex,
    QVector< Note > & notes,
    QHash< QString, QStringList > & tagNamesByNoteLocalUid,
    ErrorString & errorDescription ) const
```

importEnex reads the content of input ENEX file and converts it into a set of notes and tag names.

## Parameters

<i>enex</i>	The input ENEX file contents
<i>notes</i>	Notes read from the ENEX
<i>tagNamesByNoteLocalUid</i>	Tag names per each read note; it is the responsibility of the method caller to find the actual tags corresponding to these names and set the tag local uids and/or guids to the note
<i>errorDescription</i>	The textual description of the error if the ENEX file could not be read and converted into a set of notes and tag names for them

## Returns

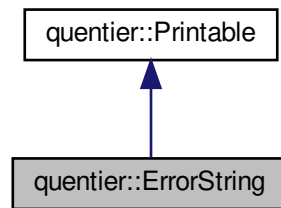
True if the ENEX file was read and converted into a set of notes and tag names successfully, false otherwise

## 4.17 quantier::ErrorString Class Reference

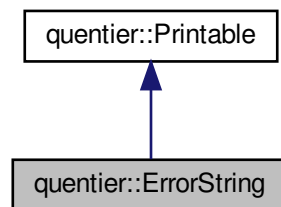
The [ErrorString](#) class encapsulates two (or more) strings which are meant to contain translatable (base) and non-translatable (details) parts of the error description.

```
#include <ErrorString.h>
```

Inheritance diagram for `quentier::ErrorString`:



Collaboration diagram for `quentier::ErrorString`:



## Public Member Functions

- **ErrorString** (const char \*error=nullptr)
- **ErrorString** (const QString &error)
- **ErrorString** (const [ErrorString](#) &other)
- [ErrorString](#) & **operator=** (const [ErrorString](#) &other)
- const QString & **base** () const
- QString & **base** ()
- const QStringList & **additionalBases** () const
- QStringList & **additionalBases** ()
- const QString & **details** () const
- QString & **details** ()
- void **setBase** (const QString &error)
- void **setBase** (const char \*error)
- void **appendBase** (const QString &error)
- void **appendBase** (const QStringList &errors)
- void **appendBase** (const char \*error)
- void **setDetails** (const QString &error)
- void **setDetails** (const char \*error)
- bool **isEmpty** () const
- void **clear** ()
- QString **localizedString** () const
- QString **nonLocalizedString** () const
- virtual QTextStream & **print** (QTextStream &strm) const override

## Additional Inherited Members

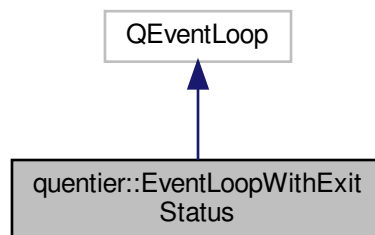
### 4.17.1 Detailed Description

The [ErrorString](#) class encapsulates two (or more) strings which are meant to contain translatable (base) and non-translatable (details) parts of the error description.

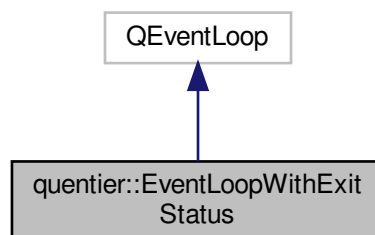
1. `base()` methods return const and non-const links to the primary translatable string
2. `details()` methods return const and non-const links to non-translatable string (coming from some third party library etc)
3. `additionalBases()` methods return const and non-const links to additional translatable strings; one translatable string is not always enough because the error message might be composed from different parts

## 4.18 `quentier::EventLoopWithExitStatus` Class Reference

Inheritance diagram for `quentier::EventLoopWithExitStatus`:



Collaboration diagram for `quentier::EventLoopWithExitStatus`:



## Public Types

- enum class **ExitStatus** { **Success** , **Failure** , **Timeout** }

## Public Slots

- void **exitAsSuccess** ()
- void **exitAsFailure** ()
- void **exitAsFailureWithError** (QString errorDescription)
- void **exitAsFailureWithErrorString** ([ErrorString](#) errorDescription)
- void **exitAsTimeout** ()

## Public Member Functions

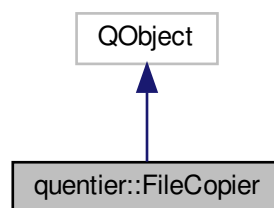
- EventLoopWithExitStatus** (QObject \*parent=nullptr)
- ExitStatus **exitStatus** () const
- const [ErrorString](#) & **errorDescription** () const

## Friends

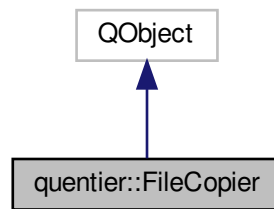
- QDebug & **operator<<** (QDebug &dbg, const ExitStatus status)
- QTextStream & **operator<<** (QTextStream &strm, const ExitStatus status)

## 4.19 quantier::FileCopier Class Reference

Inheritance diagram for quantier::FileCopier:



Collaboration diagram for quantier::FileCopier:



## Public Types

- enum class **State** { **Idle** = 0 , **Copying** , **Cancelling** }

## Public Slots

- void **copyFile** (QString sourcePath, QString destPath)
- void **cancel** ()

## Signals

- void **progressUpdate** (double progress)
- void **finished** (QString sourcePath, QString destPath)
- void **cancelled** (QString sourcePath, QString destPath)
- void **notifyError** ([ErrorString](#) error)

## Public Member Functions

- **FileCopier** (QObject \*parent=nullptr)
- State **state** () const
- QString **sourceFilePath** () const
- QString **destinationFilePath** () const
- double **currentProgress** () const

## Friends

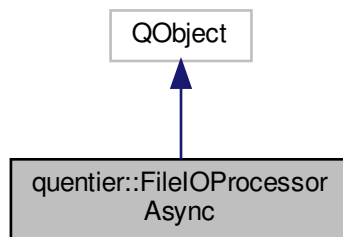
- QDebug & **operator**<< (QDebug &dbg, const State state)
- QTextStream & **operator**<< (QTextStream &strm, const State state)

## 4.20 quantier::FileIOProcessorAsync Class Reference

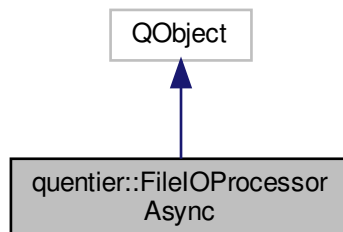
The [FileIOProcessorAsync](#) class is a wrapper under simple file IO operations, it is meant to be used for simple asynchronous IO.

```
#include <FileIOProcessorAsync.h>
```

Inheritance diagram for quantier::FileIOProcessorAsync:



Collaboration diagram for quantier::FileIOProcessorAsync:



### Public Slots

- void [onWriteFileRequest](#) (QString absoluteFilePath, QByteArray data, QUuid requestId, bool append)  
*onWriteFileRequest slot processes file write requests with given request ids*
- void [onReadFileRequest](#) (QString absoluteFilePath, QUuid requestId)  
*onReadFileRequest slot processes file read requests with given request ids*



## Signals

- void `readyForIO` ()  
*readyForIO signal is emitted when the queue for file IO is empty for some time (30 seconds by default, can also be configured via `setIdleTimePeriod` method) after the last IO event to signal listeners that they can perform some IO via the `FileIOProcessorAsync`*
- void `writeFileRequestProcessed` (bool success, `ErrorString` errorDescription, QUuid requestId)  
*writeFileRequestProcessed signal is emitted when the file write request with given id is finished*
- void `readFileRequestProcessed` (bool success, `ErrorString` errorDescription, QByteArray data, QUuid requestId)  
*readFileRequestProcessed signal is emitted when the file read request with given id is finished*

## Public Member Functions

- `FileIOProcessorAsync` (QObject \*parent=nullptr)
- void `setIdleTimePeriod` (qint32 seconds)  
*setIdleTimePeriod sets time period defining the idle state of `FileIOProcessorAsync`: once the time measured since the last IO operation is over the specified number of seconds, the class emits readyForIO signal to any interested listeners of this event. If this method is not called ever, the default idle time period would be 30 seconds.*

### 4.20.1 Detailed Description

The `FileIOProcessorAsync` class is a wrapper under simple file IO operations, it is meant to be used for simple asynchronous IO.

### 4.20.2 Member Function Documentation

#### 4.20.2.1 onReadFileRequest

```
void quantier::FileIOProcessorAsync::onReadFileRequest (
    QString absoluteFilePath,
    QUuid requestId ) [slot]
```

onReadFileRequest slot processes file read requests with given request ids

##### Parameters

<i>absoluteFilePath</i>	Absolute file path to be read
<i>requestId</i>	Unique identifier of the file read request

#### 4.20.2.2 onWriteFileRequest

```
void quantier::FileIOProcessorAsync::onWriteFileRequest (
    QString absoluteFilePath,
```

```

    QByteArray data,
    QUuid requestId,
    bool append ) [slot]

```

onWriteFileRequest slot processes file write requests with given request ids

#### Parameters

<i>absoluteFilePath</i>	Absolute file path to be written
<i>data</i>	Data to be written to the file
<i>requestId</i>	Unique identifier of the file write request
<i>append</i>	If true, the data would be appended to file, otherwise the entire file would be erased before with the data is written

#### 4.20.2.3 readFileRequestProcessed

```

void quentier::FileIOProcessorAsync::readFileRequestProcessed (
    bool success,
    ErrorString errorDescription,
    QByteArray data,
    QUuid requestId ) [signal]

```

readFileRequestProcessed signal is emitted when the file read request with given id is finished

#### Parameters

<i>success</i>	True if read operation was successful, false otherwise
<i>errorDescription</i>	Textual description of the error
<i>data</i>	Data read from file
<i>requestId</i>	Unique identifier of the file read request

#### 4.20.2.4 setIdleTimePeriod()

```

void quentier::FileIOProcessorAsync::setIdleTimePeriod (
    qint32 seconds )

```

setIdleTimePeriod sets time period defining the idle state of [FileIOProcessorAsync](#): once the time measured since the last IO operation is over the specified number of seconds, the class emits readyForIO signal to any interested listeners of this event. If this method is not called ever, the default idle time period would be 30 seconds.

#### Parameters

<i>seconds</i>	Number of seconds for idle time period
----------------	--

#### 4.20.2.5 writeFileRequestProcessed

```
void quantier::FileIOProcessorAsync::writeFileRequestProcessed (
    bool success,
    QString errorDescription,
    QUuid requestId ) [signal]
```

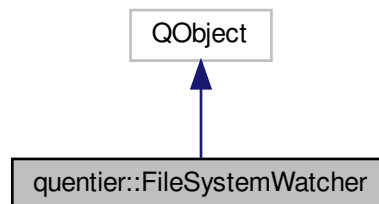
writeFileRequestProcessed signal is emitted when the file write request with given id is finished

##### Parameters

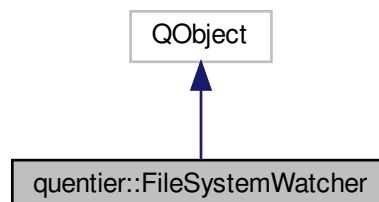
<i>success</i>	True if write operation was successful, false otherwise
<i>errorDescription</i>	Textual description of the error
<i>requestId</i>	Unique identifier of the file write request

## 4.21 quantier::FileSystemWatcher Class Reference

Inheritance diagram for quantier::FileSystemWatcher:



Collaboration diagram for quantier::FileSystemWatcher:



## Signals

- void **directoryChanged** (const QString &path)
- void **directoryRemoved** (const QString &path)
- void **fileChanged** (const QString &path)
- void **fileRemoved** (const QString &path)

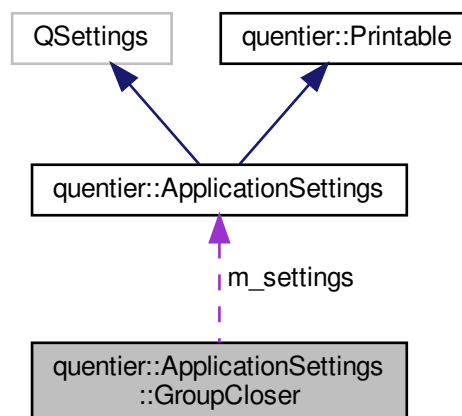
## Public Member Functions

- **FileSystemWatcher** (const int removalTimeoutMSec=FILE\_SYSTEM\_WATCHER\_DEFAULT\_REMOVAL\_TIMEOUT\_MSEC, QObject \*parent=nullptr)
- **FileSystemWatcher** (const QStringList &paths, const int removalTimeoutMSec=FILE\_SYSTEM\_WATCHER\_DEFAULT\_REMOVAL\_TIMEOUT\_MSEC, QObject \*parent=nullptr)
- void **addPath** (const QString &path)
- void **addPaths** (const QStringList &paths)
- QStringList **directories** () const
- QStringList **files** () const
- void **removePath** (const QString &path)
- void **removePaths** (const QStringList &paths)

## 4.22 quantier::ApplicationSettings::GroupCloser Struct Reference

```
#include <ApplicationSettings.h>
```

Collaboration diagram for quantier::ApplicationSettings::GroupCloser:



## Public Member Functions

- **GroupCloser** ([ApplicationSettings](#) &settings)

## Public Attributes

- [ApplicationSettings](#) & `m_settings`

### 4.22.1 Detailed Description

Helper struct for RAIL style of ensuring the group once opened would be closed even if exception is thrown after beginning the group

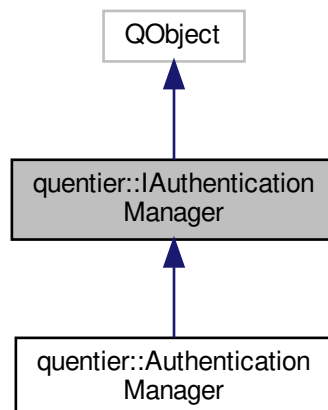
## 4.23 quantier::HTMLCleaner Class Reference

### Public Member Functions

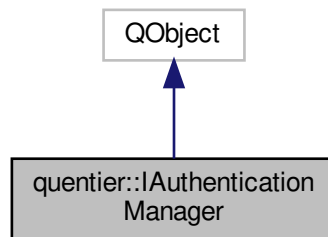
- bool **htmlToXml** (const QString &html, QString &output, QString &errorDescription)
- bool **htmlToXhtml** (const QString &html, QString &output, QString &errorDescription)
- bool **cleanupHtml** (QString &html, QString &errorDescription)

## 4.24 quantier::IAuthenticationManager Class Reference

Inheritance diagram for quantier::IAuthenticationManager:



Collaboration diagram for `quentier::IAuthenticationManager`:



### Public Slots

- virtual void **onAuthenticationRequest** ()=0

### Signals

- void **sendAuthenticationResult** (bool success, `qevercloud::UserID` userId, `QString` authToken, `qevercloud::Timestamp` authTokenExpirationTime, `QString` shardId, `QString` noteStoreUrl, `QString` webApiUrlPrefix, `QList< QNetworkCookie >` userStoreCookies, [ErrorString](#) errorDescription)

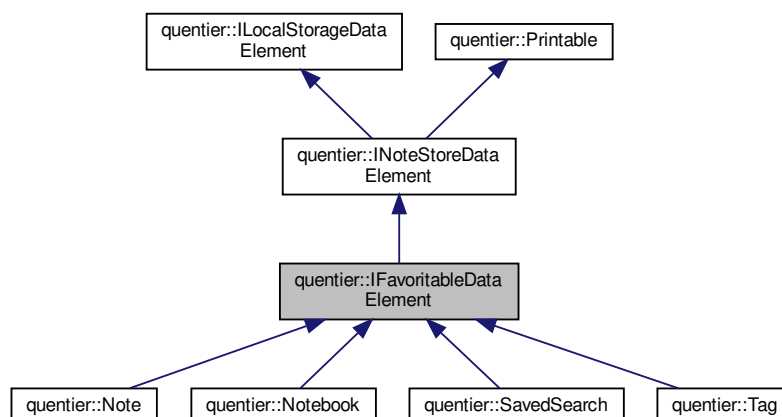
### Protected Member Functions

- **IAuthenticationManager** (`QObject *parent=nullptr`)

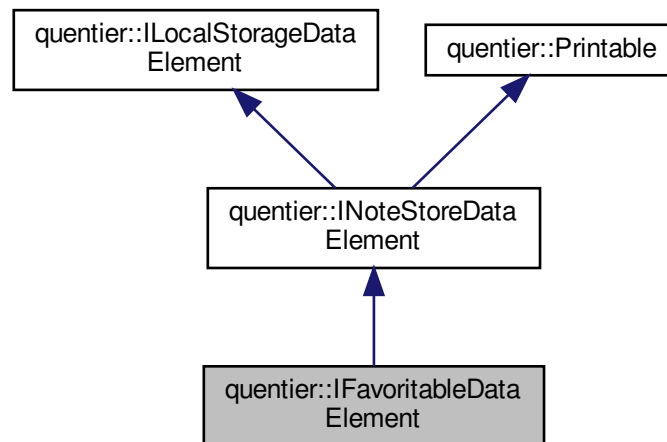
## 4.25 quentier::IFavoritableDataElement Class Reference

```
#include <IFavoritableDataElement.h>
```

Inheritance diagram for `quentier::IFavoritableDataElement`:



Collaboration diagram for quantier::IFavoritableDataElement:



### Public Member Functions

- virtual bool **isFavorited** () const =0
- virtual void **setFavorited** (const bool favored)=0

### Additional Inherited Members

#### 4.25.1 Detailed Description

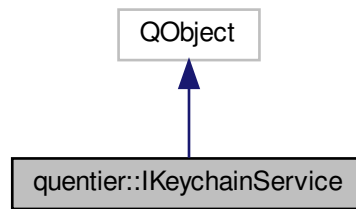
Class adding one more attribute to each note store data element subclassing it: "favorited" flag. Favorited data elements are expected to be somehow arranged for quick access in the client application's UI.

## 4.26 quantier::IKeychainService Class Reference

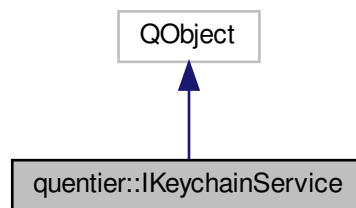
The [IKeychainService](#) interface provides methods intended to start potentially asynchronous interaction with the keychain and signals intended to notify listeners about the completion of asynchronous interactions.

```
#include <IKeychainService.h>
```

Inheritance diagram for `quentier::IKeychainService`:



Collaboration diagram for `quentier::IKeychainService`:



## Public Types

- enum class `ErrorCode` {  
    [NoError](#) , [EntryNotFound](#) , [CouldNotDeleteEntry](#) , [AccessDeniedByUser](#) ,  
    [AccessDenied](#) , [NoBackendAvailable](#) , [NotImplemented](#) , [OtherError](#) }

## Signals

- void [writePasswordJobFinished](#) (QUuid requestId, [ErrorCode](#) errorCode, [ErrorString](#) errorDescription)
- void [readPasswordJobFinished](#) (QUuid requestId, [ErrorCode](#) errorCode, [ErrorString](#) errorDescription, QString password)
- void [deletePasswordJobFinished](#) (QUuid requestId, [ErrorCode](#) errorCode, [ErrorString](#) errorDescription)

## Public Member Functions

- virtual QUuid [startWritePasswordJob](#) (const QString &service, const QString &key, const QString &password)=0
- virtual QUuid [startReadPasswordJob](#) (const QString &service, const QString &key)=0
- virtual QUuid [startDeletePasswordJob](#) (const QString &service, const QString &key)=0



## Protected Member Functions

- **IKeychainService** (QObject \*parent=nullptr)

## Friends

- QTextStream & **operator**<< (QTextStream &strm, const [ErrorCode](#) errorCode)
- QDebug & **operator**<< (QDebug &dbg, const [ErrorCode](#) errorCode)

### 4.26.1 Detailed Description

The [IKeychainService](#) interface provides methods intended to start potentially asynchronous interaction with the keychain and signals intended to notify listeners about the completion of asynchronous interactions.

### 4.26.2 Member Enumeration Documentation

#### 4.26.2.1 ErrorCode

```
enum quentier::IKeychainService::ErrorCode [strong]
```

Error codes for results of operations with the keychain service

##### Enumerator

NoError	No error occurred, operation was successful
EntryNotFound	For the given key no data was found
CouldNotDeleteEntry	Could not delete existing secret data
AccessDeniedByUser	<a href="#">User</a> denied access to keychain
AccessDenied	Access denied for some reason
NoBackendAvailable	No platform-specific keychain service available
NotImplemented	Not implemented on platform
OtherError	Something else went wrong, the error description specifies what

### 4.26.3 Member Function Documentation

#### 4.26.3.1 deletePasswordJobFinished

```
void quentier::IKeychainService::deletePasswordJobFinished (
    QUuid requestId,
```

```
    ErrorCode errorCode,  
    ErrorString errorDescription )    [signal]
```

deletePasswordJobFinished signal should be emitted in response to the call of startDeletePasswordJob method

## Parameters

<i>requestId</i>	Request id returned from startDeletePasswordJob method
<i>errorCode</i>	Error code determining whether the operation was successful or some error has occurred
<i>errorDescription</i>	Textual description of error in case of unsuccessful execution

### 4.26.3.2 readPasswordJobFinished

```
void quotienter::IKeychainService::readPasswordJobFinished (
    QUuid requestId,
    ErrorCode errorCode,
    ErrorString errorDescription,
    QString password ) [signal]
```

readPasswordJobFinished signal should be emitted in response to the call of startReadPasswordJob method

## Parameters

<i>requestId</i>	Request id returned from startReadPasswordJob method
<i>errorCode</i>	Error code determining whether the operation was successful or some error has occurred
<i>errorDescription</i>	Textual description of error in case of unsuccessful execution
<i>password</i>	Password read from the keychain

### 4.26.3.3 startDeletePasswordJob()

```
virtual QUuid quotienter::IKeychainService::startDeletePasswordJob (
    const QString & service,
    const QString & key ) [pure virtual]
```

startDeletePasswordJob slot should start the potentially asynchronous process of deleting the password from the keychain. When ready, this slot is expected to emit deletePasswordJobFinished signal.

## Parameters

<i>service</i>	Name of service within the keychain
<i>key</i>	Key under which the password is stored

## Returns

Unique identifier assigned to this delete password request

#### 4.26.3.4 startReadPasswordJob()

```
virtual QUuid quentier::IKeychainService::startReadPasswordJob (
    const QString & service,
    const QString & key ) [pure virtual]
```

startReadPasswordJob slot should start the potentially asynchronous process of reading the password from the keychain. When ready, this slot is expected to emit readPasswordJobFinished signal.

##### Parameters

<i>service</i>	Name of service within the keychain
<i>key</i>	Key under which the password is stored

##### Returns

Unique identifier assigned to this read password request

#### 4.26.3.5 startWritePasswordJob()

```
virtual QUuid quentier::IKeychainService::startWritePasswordJob (
    const QString & service,
    const QString & key,
    const QString & password ) [pure virtual]
```

startWritePasswordJob slot should start the potentially asynchronous process of storing the password in the keychain. When ready, this slot is expected to emit writePasswordJobFinished signal.

##### Parameters

<i>service</i>	Name of service within the keychain
<i>key</i>	Key to store the password under
<i>password</i>	Password to store in the keychain

##### Returns

Unique identifier assigned to this write password request

#### 4.26.3.6 writePasswordJobFinished

```
void quentier::IKeychainService::writePasswordJobFinished (
    QUuid requestId,
    ErrorCode errorCode,
    ErrorString errorDescription ) [signal]
```

writePasswordJobFinished signal should be emitted in response to the call of startWritePasswordJob method

## Parameters

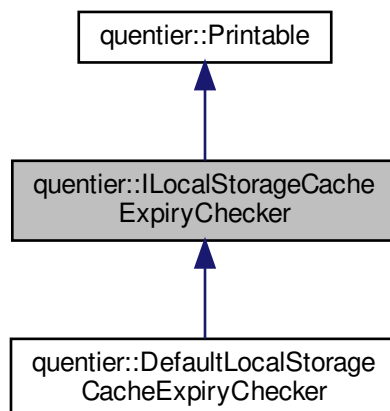
<i>requestId</i>	Request id returned from startWritePasswordJob method
<i>errorCode</i>	Error code determining whether the operation was successful or some error has occurred
<i>errorDescription</i>	Textual description of error in case of unsuccessful execution

## 4.27 quantier::ILocalStorageCacheExpiryChecker Class Reference

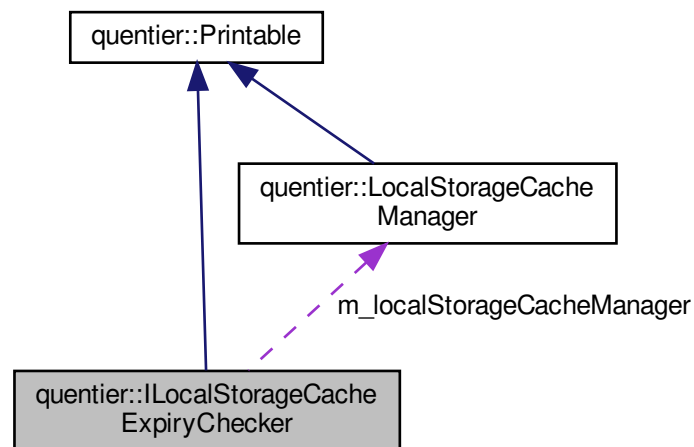
The [ILocalStorageCacheExpiryChecker](#) class represents the interface for cache expiry checker used by [LocalStorageCacheManager](#) to see whether particular caches (of notes, notebooks, tags, linked notebooks and/or saved searches) need to be shrunk.

```
#include <ILocalStorageCacheExpiryChecker.h>
```

Inheritance diagram for quantier::ILocalStorageCacheExpiryChecker:



Collaboration diagram for `quentier::ILocalStorageCacheExpiryChecker`:



## Public Member Functions

- virtual [ILocalStorageCacheExpiryChecker](#) \* [clone](#) () const =0
- virtual bool [checkNotes](#) () const =0
- virtual bool [checkResources](#) () const =0
- virtual bool [checkNotebooks](#) () const =0
- virtual bool [checkTags](#) () const =0
- virtual bool [checkLinkedNotebooks](#) () const =0
- virtual bool [checkSavedSearches](#) () const =0
- virtual QTextStream & [print](#) (QTextStream &strm) const =0

*Print the internal information about [ILocalStorageCacheExpiryChecker](#) implementation instance to the text stream.*

## Protected Member Functions

- [ILocalStorageCacheExpiryChecker](#) (const [LocalStorageCacheManager](#) &cacheManager)

## Protected Attributes

- const [LocalStorageCacheManager](#) & [m\\_localStorageCacheManager](#)

### 4.27.1 Detailed Description

The [ILocalStorageCacheExpiryChecker](#) class represents the interface for cache expiry checker used by [LocalStorageCacheManager](#) to see whether particular caches (of notes, notebooks, tags, linked notebooks and/or saved searches) need to be shrunk.

## 4.27.2 Member Function Documentation

### 4.27.2.1 checkLinkedNotebooks()

```
virtual bool quentier::ILocalStorageCacheExpiryChecker::checkLinkedNotebooks ( ) const [pure virtual]
```

#### Returns

False if the cache of linked notebooks needs to be shrunk (due to its size or whatever other reason), true otherwise

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

### 4.27.2.2 checkNotebooks()

```
virtual bool quentier::ILocalStorageCacheExpiryChecker::checkNotebooks ( ) const [pure virtual]
```

#### Returns

False if the cache of notebooks needs to be shrunk (due to its size or whatever other reason), true otherwise

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

### 4.27.2.3 checkNotes()

```
virtual bool quentier::ILocalStorageCacheExpiryChecker::checkNotes ( ) const [pure virtual]
```

#### Returns

False if the cache of notes needs to be shrunk (due to its size or whatever other reason), true otherwise

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

### 4.27.2.4 checkResources()

```
virtual bool quentier::ILocalStorageCacheExpiryChecker::checkResources ( ) const [pure virtual]
```

#### Returns

False if the cache of resources needs to be shrunk (due to its size or whatever other reason), true otherwise

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

#### 4.27.2.5 checkSavedSearches()

```
virtual bool quentier::ILocalStorageCacheExpiryChecker::checkSavedSearches ( ) const [pure virtual]
```

##### Returns

False if the cache of saved searches needs to be shrunk (due to its size or whatever other reason), true otherwise

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

#### 4.27.2.6 checkTags()

```
virtual bool quentier::ILocalStorageCacheExpiryChecker::checkTags ( ) const [pure virtual]
```

##### Returns

False if the cache of tags needs to be shrunk (due to its size or whatever other reason), true otherwise

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

#### 4.27.2.7 clone()

```
virtual ILocalStorageCacheExpiryChecker* quentier::ILocalStorageCacheExpiryChecker::clone ( ) const [pure virtual]
```

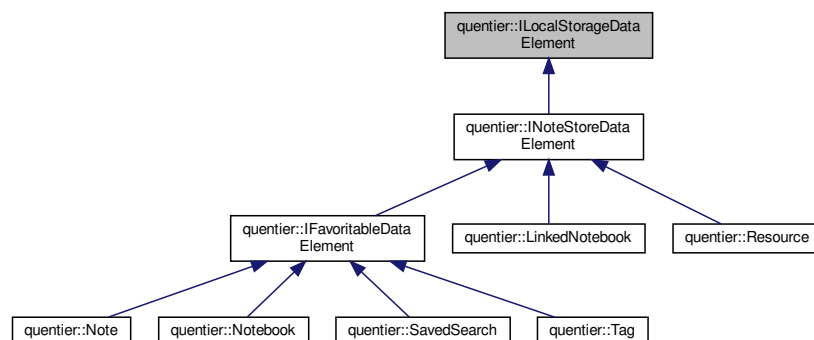
##### Returns

A pointer to the newly allocated copy of a particular [ILocalStorageCacheExpiryChecker](#) implementation

Implemented in [quentier::DefaultLocalStorageCacheExpiryChecker](#).

## 4.28 quentier::ILocalStorageDataElement Class Reference

Inheritance diagram for quentier::ILocalStorageDataElement:





## Public Member Functions

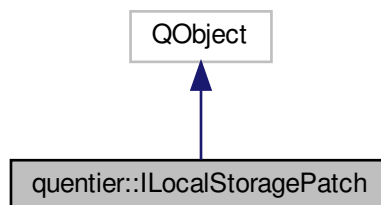
- virtual const QString **localUid** () const =0
- virtual void **setLocalUid** (const QString &guid)=0
- virtual void **unsetLocalUid** ()=0

## 4.29 quantier::ILocalStoragePatch Class Reference

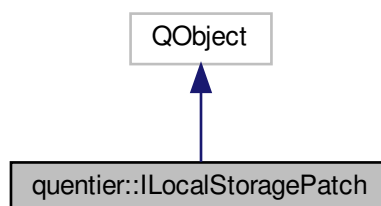
The [ILocalStoragePatch](#) class represents the interface for patches of local storage. Each such patch somehow changes the layout of local storage persistence so that only compliant & corresponding versions of libquantier can be used to work with it.

```
#include <ILocalStoragePatch.h>
```

Inheritance diagram for quantier::ILocalStoragePatch:



Collaboration diagram for quantier::ILocalStoragePatch:



## Signals

- void [progress](#) (double progress)
- void [backupProgress](#) (double progress)
- void [restoreBackupProgress](#) (double progress)

## Public Member Functions

- virtual int [fromVersion](#) () const =0
- virtual int [toVersion](#) () const =0
- virtual QString [patchShortDescription](#) () const =0
- virtual QString [patchLongDescription](#) () const =0
- virtual bool [backupLocalStorage](#) (ErrorString &errorDescription)=0
- virtual bool [restoreLocalStorageFromBackup](#) (ErrorString &errorDescription)=0
- virtual bool [removeLocalStorageBackup](#) (ErrorString &errorDescription)=0
- virtual bool [apply](#) (ErrorString &errorDescription)=0

## Protected Member Functions

- [ILocalStoragePatch](#) (QObject \*parent=nullptr)

## Friends

- class [LocalStorageDatabaseUpgrader](#)

### 4.29.1 Detailed Description

The [ILocalStoragePatch](#) class represents the interface for patches of local storage. Each such patch somehow changes the layout of local storage persistence so that only compliant & corresponding versions of libquentier can be used to work with it.

### 4.29.2 Member Function Documentation

#### 4.29.2.1 [apply\(\)](#)

```
virtual bool quentier::ILocalStoragePatch::apply (
    ErrorString & errorDescription ) [pure virtual]
```

Apply the patch to local storage

#### Parameters

<i>errorDescription</i>	The textual description of the error in case of patch application failure
-------------------------	---

#### Returns

True in case of successful patch application, false otherwise

#### 4.29.2.2 backupLocalStorage()

```
virtual bool quentier::ILocalStoragePatch::backupLocalStorage (
    ErrorString & errorDescription ) [pure virtual]
```

Backup either the entire local storage or its parts affected by the particular patch, should be called before applying the patch (but can be skipped if not desired).

##### Parameters

<i>errorDescription</i>	The textual description of the error in case of backup preparation failure
-------------------------	--

##### Returns

True if the local storage was backed up for the patch successfully, false otherwise

#### 4.29.2.3 backupProgress

```
void quentier::ILocalStoragePatch::backupProgress (
    double progress ) [signal]
```

Local storage backup preparation progress

##### Parameters

<i>progress</i>	Backup preparation progress value, from 0 to 1
-----------------	--

#### 4.29.2.4 fromVersion()

```
virtual int quentier::ILocalStoragePatch::fromVersion ( ) const [pure virtual]
```

##### Returns

Version of local storage to which the patch needs to be applied

#### 4.29.2.5 patchLongDescription()

```
virtual QString quentier::ILocalStoragePatch::patchLongDescription ( ) const [pure virtual]
```

##### Returns

Long i.e. detailed description of the patch

#### 4.29.2.6 patchShortDescription()

```
virtual QString quentier::ILocalStoragePatch::patchShortDescription ( ) const [pure virtual]
```

##### Returns

Short description of the patch

#### 4.29.2.7 progress

```
void quentier::ILocalStoragePatch::progress (
    double progress ) [signal]
```

Patch application progress signal

##### Parameters

<i>progress</i>	Patch application progress value, from 0 to 1
-----------------	---

#### 4.29.2.8 removeLocalStorageBackup()

```
virtual bool quentier::ILocalStoragePatch::removeLocalStorageBackup (
    QString & errorDescription ) [pure virtual]
```

Remove the previously made backup of local storage, presumably after successful application of the patch so the backup is no longer needed. It won't work if no backup was made before applying a patch, obviously.

##### Parameters

<i>errorDescription</i>	The textual description of the error in case of failure to remove the local storage backup
-------------------------	--

##### Returns

True if local storage backup was successfully removed, false otherwise

#### 4.29.2.9 restoreBackupProgress

```
void quentier::ILocalStoragePatch::restoreBackupProgress (
    double progress ) [signal]
```

Local storage restoration from backup progress

## Parameters

<i>progress</i>	Local storage restoration from backup progress value, from 0 to 1
-----------------	---

**4.29.2.10 restoreLocalStorageFromBackup()**

```
virtual bool quantier::ILocalStoragePatch::restoreLocalStorageFromBackup (
    ErrorString & errorDescription ) [pure virtual]
```

Restore local storage from previously made backup, presumably after the failed attempt to apply a patch. Won't work if no backup was made before applying a patch, obviously.

## Parameters

<i>errorDescription</i>	The textual description of the error in case of failure to restore the local storage from backup
-------------------------	--

## Returns

True if local storage was successfully restored from backup, false otherwise

**4.29.2.11 toVersion()**

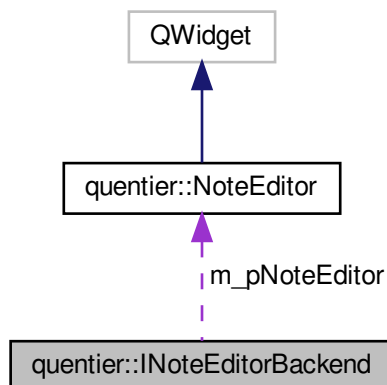
```
virtual int quantier::ILocalStoragePatch::toVersion ( ) const [pure virtual]
```

## Returns

Version of local storage to which the patch would upgrade the local storage

## 4.30 quantier::INoteEditorBackend Class Reference

Collaboration diagram for quantier::INoteEditorBackend:



### Public Types

- enum class **Rotation** { **Clockwise** = 0 , **Counterclockwise** }

### Public Member Functions

- virtual void **initialize** ([LocalStorageManagerAsync](#) &localStorageManager, [SpellChecker](#) &spellChecker, const [Account](#) &account, QThread \*pBackgroundJobsThread)=0
- virtual QObject \* **object** ()=0
- virtual QWidget \* **widget** ()=0
- virtual void **setAccount** (const [Account](#) &account)=0
- virtual void **setUndoStack** (QUndoStack \*pUndoStack)=0
- virtual void **setInitialPageHtml** (const QString &html)=0
- virtual void **setNoteNotFoundPageHtml** (const QString &html)=0
- virtual void **setNoteDeletedPageHtml** (const QString &html)=0
- virtual void **setNoteLoadingPageHtml** (const QString &html)=0
- virtual bool **isNoteLoaded** () const =0
- virtual qint64 **idleTime** () const =0
- virtual void **convertToNote** ()=0
- virtual void **saveNoteToLocalStorage** ()=0
- virtual void **setNoteTitle** (const QString &noteTitle)=0
- virtual void **setTagIds** (const QStringList &tagLocalUids, const QStringList &tagGuids)=0
- virtual void **undo** ()=0
- virtual void **redo** ()=0

- virtual void **cut** ()=0
- virtual void **copy** ()=0
- virtual void **paste** ()=0
- virtual void **pasteUnformatted** ()=0
- virtual void **selectAll** ()=0
- virtual void **formatSelectionAsSourceCode** ()=0
- virtual void **fontMenu** ()=0
- virtual void **textBold** ()=0
- virtual void **textItalic** ()=0
- virtual void **textUnderline** ()=0
- virtual void **textStrikethrough** ()=0
- virtual void **textHighlight** ()=0
- virtual void **alignLeft** ()=0
- virtual void **alignCenter** ()=0
- virtual void **alignRight** ()=0
- virtual void **alignFull** ()=0
- virtual QString **selectedText** () const =0
- virtual bool **hasSelection** () const =0
- virtual void **findNext** (const QString &text, const bool matchCase) const =0
- virtual void **findPrevious** (const QString &text, const bool matchCase) const =0
- virtual void **replace** (const QString &textToReplace, const QString &replacementText, const bool match↵Case)=0
- virtual void **replaceAll** (const QString &textToReplace, const QString &replacementText, const bool match↵Case)=0
- virtual void **insertToDoCheckbox** ()=0
- virtual void **insertInAppNoteLink** (const QString &userId, const QString &shardId, const QString &noteGuid, const QString &linkText)=0
- virtual void **setSpellcheck** (const bool enabled)=0
- virtual bool **spellCheckEnabled** () const =0
- virtual void **setFont** (const QFont &font)=0
- virtual void **setFontHeight** (const int height)=0
- virtual void **setFontColor** (const QColor &color)=0
- virtual void **setBackgroundColor** (const QColor &color)=0
- virtual QPalette **defaultPalette** () const =0
- virtual void **setDefaultPalette** (const QPalette &pal)=0
- virtual const QFont \* **defaultFont** () const =0
- virtual void **setDefaultFont** (const QFont &font)=0
- virtual void **insertHorizontalLine** ()=0
- virtual void **increaseFontSize** ()=0
- virtual void **decreaseFontSize** ()=0
- virtual void **increaseIndentation** ()=0
- virtual void **decreaseIndentation** ()=0
- virtual void **insertBulletedList** ()=0
- virtual void **insertNumberedList** ()=0
- virtual void **insertTableDialog** ()=0
- virtual void **insertFixedWidthTable** (const int rows, const int columns, const int widthInPixels)=0
- virtual void **insertRelativeWidthTable** (const int rows, const int columns, const double relativeWidth)=0
- virtual void **insertTableRow** ()=0
- virtual void **insertTableColumn** ()=0
- virtual void **removeTableRow** ()=0
- virtual void **removeTableColumn** ()=0
- virtual void **addAttachmentDialog** ()=0
- virtual void **saveAttachmentDialog** (const QByteArray &resourceHash)=0
- virtual void **saveAttachmentUnderCursor** ()=0
- virtual void **openAttachment** (const QByteArray &resourceHash)=0

- virtual void **openAttachmentUnderCursor** ()=0
- virtual void **copyAttachment** (const QByteArray &resourceHash)=0
- virtual void **copyAttachmentUnderCursor** ()=0
- virtual void **removeAttachment** (const QByteArray &resourceHash)=0
- virtual void **removeAttachmentUnderCursor** ()=0
- virtual void **renameAttachment** (const QByteArray &resourceHash)=0
- virtual void **renameAttachmentUnderCursor** ()=0
- virtual void **rotateImageAttachment** (const QByteArray &resourceHash, const Rotation rotationDirection)=0
- virtual void **rotateImageAttachmentUnderCursor** (const Rotation rotationDirection)=0
- virtual void **encryptSelectedText** ()=0
- virtual void **decryptEncryptedTextUnderCursor** ()=0
- virtual void **decryptEncryptedText** (QString encryptedText, QString cipher, QString keyLength, QString hint, QString enCryptIndex)=0
- virtual void **hideDecryptedTextUnderCursor** ()=0
- virtual void **hideDecryptedText** (QString encryptedText, QString decryptedText, QString cipher, QString keyLength, QString hint, QString enDecryptedIndex)=0
- virtual void **editHyperlinkDialog** ()=0
- virtual void **copyHyperlink** ()=0
- virtual void **removeHyperlink** ()=0
- virtual void **onNoteLoadCancelled** ()=0
- virtual bool **print** (QPrinter &printer, [ErrorString](#) &errorDescription)=0
- virtual bool **exportToPdf** (const QString &absoluteFilePath, [ErrorString](#) &errorDescription)=0
- virtual bool **exportToEnex** (const QStringList &tagNames, QString &enex, [ErrorString](#) &errorDescription)=0
- virtual QString **currentNoteLocalUid** () const =0
- virtual void **setCurrentNoteLocalUid** (const QString &noteLocalUid)=0
- virtual void **clear** ()=0
- virtual bool **isModified** () const =0
- virtual bool **isEditorPageModified** () const =0
- virtual void **setFocusToEditor** ()=0

## Protected Member Functions

- **INoteEditorBackend** ([NoteEditor](#) \*parent)

## Protected Attributes

- [NoteEditor](#) \* **m\_pNoteEditor**

## Friends

- QUENTIER\_EXPORT QTextStream & **operator<<** (QTextStream &strm, const Rotation rotation)
- QUENTIER\_EXPORT QDebug & **operator<<** (QDebug &dbg, const Rotation rotation)

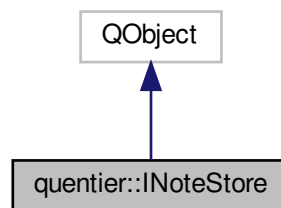


## 4.31 quantier::INoteStore Class Reference

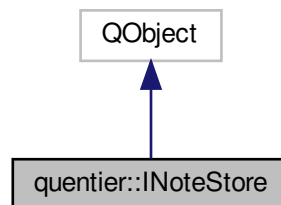
[INoteStore](#) is the interface which provides methods required for the implementation of NoteStore part of Evernote EDAM sync protocol.

```
#include <INoteStore.h>
```

Inheritance diagram for quantier::INoteStore:



Collaboration diagram for quantier::INoteStore:



### Signals

- void **getNoteAsyncFinished** (qint32 errorCode, qevercloud::Note note, qint32 rateLimitSeconds, [ErrorString](#) errorDescription)
- void **getResourceAsyncFinished** (qint32 errorCode, qevercloud::Resource resource, qint32 rateLimitSeconds, [ErrorString](#) errorDescription)

### Public Member Functions

- virtual [INoteStore](#) \* **create** () const =0
- virtual QString **noteStoreUrl** () const =0
- virtual void **setNoteStoreUrl** (QString [noteStoreUrl](#))=0

- virtual void [setAuthData](#) (QString authenticationToken, QList< QNetworkCookie > cookies)=0
- virtual void [stop](#) ()=0
- virtual qint32 [createNotebook](#) ([Notebook](#) &notebook, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds, QString linkedNotebookAuthToken=())=0
- virtual qint32 [updateNotebook](#) ([Notebook](#) &notebook, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds, QString linkedNotebookAuthToken=())=0
- virtual qint32 [createNote](#) ([Note](#) &note, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds, QString linkedNotebookAuthToken=())=0
- virtual qint32 [updateNote](#) ([Note](#) &note, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds, QString linkedNotebookAuthToken=())=0
- virtual qint32 [createTag](#) ([Tag](#) &tag, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds, QString linkedNotebookAuthToken=())=0
- virtual qint32 [updateTag](#) ([Tag](#) &tag, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds, QString linkedNotebookAuthToken=())=0
- virtual qint32 [createSavedSearch](#) ([SavedSearch](#) &savedSearch, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [updateSavedSearch](#) ([SavedSearch](#) &savedSearch, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [getSyncState](#) (qevercloud::SyncState &syncState, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [getSyncChunk](#) (const qint32 afterUSN, const qint32 maxEntries, const qevercloud::SyncChunkFilter &filter, qevercloud::SyncChunk &syncChunk, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [getLinkedNotebookSyncState](#) (const qevercloud::LinkedNotebook &linkedNotebook, const QString &authToken, qevercloud::SyncState &syncState, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [getLinkedNotebookSyncChunk](#) (const qevercloud::LinkedNotebook &linkedNotebook, const qint32 afterUSN, const qint32 maxEntries, const QString &linkedNotebookAuthToken, const bool fullSyncOnly, qevercloud::SyncChunk &syncChunk, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [getNote](#) (const bool withContent, const bool withResourcesData, const bool withResourcesRecognition, const bool withResourceAlternateData, [Note](#) &note, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual bool [getNoteAsync](#) (const bool withContent, const bool withResourceData, const bool withResourcesRecognition, const bool withResourceAlternateData, const bool withSharedNotes, const bool withNoteAppDataValues, const bool withResourceAppDataValues, const bool withNoteLimits, const QString &noteGuid, const QString &authToken, [ErrorString](#) &errorDescription)=0
- virtual qint32 [getResource](#) (const bool withDataBody, const bool withRecognitionDataBody, const bool withAlternateDataBody, const bool withAttributes, const QString &authToken, [Resource](#) &resource, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual bool [getResourceAsync](#) (const bool withDataBody, const bool withRecognitionDataBody, const bool withAlternateDataBody, const bool withAttributes, const QString &resourceGuid, const QString &authToken, [ErrorString](#) &errorDescription)=0
- virtual qint32 [authenticateToSharedNotebook](#) (const QString &shareKey, qevercloud::AuthenticationResult &authResult, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0

## Protected Member Functions

- **INoteStore** (QObject \*parent=nullptr)

### 4.31.1 Detailed Description

[INoteStore](#) is the interface which provides methods required for the implementation of NoteStore part of Evernote EDAM sync protocol.

## 4.31.2 Member Function Documentation

### 4.31.2.1 authenticateToSharedNotebook()

```
virtual qint32 quotientier::INoteStore::authenticateToSharedNotebook (
    const QString & shareKey,
    qevercloud::AuthenticationResult & authResult,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Authenticate to shared notebook

#### Parameters

<i>shareKey</i>	The shared notebook global identifier
<i>authResult</i>	Output parameter, the result of authentication
<i>errorDescription</i>	The textual description of the error if authentication to shared notebook could not be performed
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

#### Returns

Error code, 0 in case of successful authentication to shared notebook, other values corresponding to qevercloud::EDAMErrorCode enumeration instead

### 4.31.2.2 createNote()

```
virtual qint32 quotientier::INoteStore::createNote (
    Note & note,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds,
    QString linkedNotebookAuthToken = {} ) [pure virtual]
```

Create note

#### Parameters

<i>note</i>	<a href="#">Note</a> to be created
<i>errorDescription</i>	The textual description of the error in case note could not be created
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED
<i>linkedNotebookAuthToken</i>	If a note is created within another user's account, the corresponding auth token should be set, otherwise the note would be created in user's own account

**Returns**

Error code, 0 in case of successful note creation, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

**4.31.2.3 createNotebook()**

```
virtual qint32 qentier::INoteStore::createNotebook (
    Notebook & notebook,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds,
    QString linkedNotebookAuthToken = {} ) [pure virtual]
```

Create notebook

**Parameters**

<i>notebook</i>	<a href="#">Notebook</a> to be created, must have name set and either "active" or "default notebook" fields may be set
<i>errorDescription</i>	The textual description of the error in case notebook could not be created
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>
<i>linkedNotebookAuthToken</i>	If a notebook is created within another user's account, the corresponding auth token should be set, otherwise the notebook would be created in user's own account

**Returns**

Error code, 0 in case of successful notebook creation, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

**4.31.2.4 createSavedSearch()**

```
virtual qint32 qentier::INoteStore::createSavedSearch (
    SavedSearch & savedSearch,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Create saved search

**Parameters**

<i>savedSearch</i>	Saved search to be created, must have name and query set, can also have search scope set
<i>errorDescription</i>	The textual description of the error in case saved search could not be created
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>
Generated by Doxygen	

## Returns

Error code, 0 in case of successful saved search creation, other values corresponding to qevercloud::EDAMErrorCode enumeration instead

## 4.31.2.5 createTag()

```
virtual qint32 quentier::INoteStore::createTag (
    Tag & tag,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds,
    QString linkedNotebookAuthToken = {} ) [pure virtual]
```

## Create tag

## Parameters

<i>note</i>	<a href="#">Tag</a> to be created, must have name set, can also have parent guid set
<i>errorDescription</i>	The textual description of the error in case tag could not be created
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED
<i>linkedNotebookAuthToken</i>	If a tag is created within another user's account, the corresponding auth token should be set, otherwise the tag would be created in user's own account

## Returns

Error code, 0 in case of successful tag creation, other values corresponding to qevercloud::EDAMErrorCode enumeration instead

## 4.31.2.6 getLinkedNotebookSyncChunk()

```
virtual qint32 quentier::INoteStore::getLinkedNotebookSyncChunk (
    const qevercloud::LinkedNotebook & linkedNotebook,
    const qint32 afterUSN,
    const qint32 maxEntries,
    const QString & linkedNotebookAuthToken,
    const bool fullSyncOnly,
    qevercloud::SyncChunk & syncChunk,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

## Get linked notebook sync chunk

## Parameters

<i>linkedNotebook</i>	The linked notebook for which the sync chunk is being retrieved, must contain identifying information and permissions to access the notebook in question
<i>afterUSN</i>	The USN after which the sync chunks are being requested
<i>maxEntries</i>	Max number of items within the sync chunk to be returned
<i>linkedNotebookAuthToken</i>	The authentication token to use for the data from the linked notebook
<i>fullSyncOnly</i>	If true then client only wants initial data for a full sync. In this case Evernote service will not return any expunged objects and will not return any resources since these are also provided in their corresponding notes
<i>syncChunk</i>	Output parameter, the sync chunk
<i>errorDescription</i>	The textual description of the error in case linked notebook sync chunk could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

## Returns

Error code, 0 in case of successful linked notebook sync chunk retrieval, other values corresponding to qevercloud::EDAMErrorCode enumeration instead

## 4.31.2.7 getLinkedNotebookSyncState()

```
virtual qint32 quentier::INoteStore::getLinkedNotebookSyncState (
    const qevercloud::LinkedNotebook & linkedNotebook,
    const QString & authToken,
    qevercloud::SyncState & syncState,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Get linked notebook sync state

## Parameters

<i>linkedNotebook</i>	The linked notebook for which the sync state is being retrieved, must contain identifying information and permissions to access the notebook in question
<i>authToken</i>	The authentication token to use for the data from the linked notebook
<i>syncState</i>	Output parameter, the sync state
<i>errorDescription</i>	The textual description of the error in case linked notebook sync state could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

**Returns**

Error code, 0 in case of successful linked notebook sync state retrieval, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

**4.31.2.8 getNote()**

```
virtual qint32 quentier::INoteStore::getNote (
    const bool withContent,
    const bool withResourcesData,
    const bool withResourcesRecognition,
    const bool withResourceAlternateData,
    Note & note,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Get note synchronously

**Parameters**

<i>withContent</i>	If true, the returned note would include the content
<i>withResourcesData</i>	If true, any resources the note might have would include their full data
<i>withResourcesRecognition</i>	If true, any resources the note might have and which have Evernote supplied recognition would include their full recognition data
<i>withResourceAlternateData</i>	If true, any resources the note might have would include their full alternate data
<i>note</i>	Input and output parameter, the retrieved note, must have guid set
<i>errorDescription</i>	The textual description of the error in case the note could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>

**Returns**

Error code, 0 in case of successful note retrieval, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

**4.31.2.9 getNoteAsync()**

```
virtual bool quentier::INoteStore::getNoteAsync (
    const bool withContent,
    const bool withResourceData,
    const bool withResourcesRecognition,
    const bool withResourceAlternateData,
    const bool withSharedNotes,
    const bool withNoteAppDataValues,
    const bool withResourceAppDataValues,
```

```

    const bool withNoteLimits,
    const QString & noteGuid,
    const QString & authToken,
    ErrorString & errorDescription ) [pure virtual]

```

Get note asynchronously

If the method returned true, the actual result of the method invocation would be returned via the emission of get←→ NoteAsyncFinished signal.

#### Parameters

<i>withContent</i>	If true, the returned note would include the content
<i>withResourcesData</i>	If true, any resources the note might have would include their full data
<i>withResourcesRecognition</i>	If true, any resources the note might have and which have Evernote supplied recognition would include their full recognition data
<i>withResourceAlternateData</i>	If true, any resources the note might have would include their full alternate data
<i>withSharedNotes</i>	If true, any shared notes contained within the note would be provided along with the asynchronously fetched result
<i>withNoteAppDataValues</i>	If true, the asynchronously fetched note would contain the app data values
<i>withResourceAppDataValues</i>	If true, the resources of asynchronously fetched note would contain the app data values
<i>withNoteLimits</i>	If true, the asynchronously fetched note would contain note limits
<i>noteGuid</i>	The guid of the note to be retrieved
<i>authToken</i>	Authentication token to use for note retrieval
<i>errorDescription</i>	The textual description of the error if the launch of async note retrieval has failed

#### Returns

True if the launch of async note retrieval was successful, false otherwise

#### 4.31.2.10 getResource()

```

virtual qint32 quantier::INoteStore::getResource (
    const bool withDataBody,
    const bool withRecognitionDataBody,
    const bool withAlternateDataBody,
    const bool withAttributes,
    const QString & authToken,
    Resource & resource,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]

```

Get resource synchronously

#### Parameters

<i>withDataBody</i>	If true, the returned resource would include its data body
<i>withRecognitionDataBody</i>	If true, the returned resource would include its recognition data body



## Parameters

<i>withAlternateDataBody</i>	If true, the returned resource would include its alternate data body
<i>withAttributes</i>	If true, the returned resource would include its attributes
<i>authToken</i>	Authentication token to use for resource retrieval
<i>resource</i>	Input and output parameter, the retrieved resource, must have guid set
<i>errorDescription</i>	The textual description of the error in case the resource could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

## Returns

Error code, 0 in case of successful resource retrieval, other values corresponding to qevercloud::EDAMError↵  
Code enumeration instead

## 4.31.2.11 getResourceAsync()

```
virtual bool quantier::INoteStore::getResourceAsync (
    const bool withDataBody,
    const bool withRecognitionDataBody,
    const bool withAlternateDataBody,
    const bool withAttributes,
    const QString & resourceGuid,
    const QString & authToken,
    QString & errorDescription ) [pure virtual]
```

Get resource asynchronously

If the method returned true, the actual result of the method invocation would be returned via the emission of get↵  
ResourceAsyncFinished signal.

## Parameters

<i>withDataBody</i>	If true, the returned resource would include its data body
<i>withRecognitionDataBody</i>	If true, the returned resource would include its recognition data body
<i>withAlternateDataBody</i>	If true, the returned resource would include its alternate data body
<i>withAttributes</i>	If true, the returned resource would include its attributes
<i>resourceGuid</i>	The guid of the resource to be retrieved
<i>authToken</i>	Authentication token to use for resource retrieval
<i>errorDescription</i>	The textual description of the error if the launch of async resource retrieval has failed

## Returns

True if the launch of async resource retrieval was successful, false otherwise

**4.31.2.12 getSyncChunk()**

```
virtual qint32 quentier::INoteStore::getSyncChunk (
    const qint32 afterUSN,
    const qint32 maxEntries,
    const qevercloud::SyncChunkFilter & filter,
    qevercloud::SyncChunk & syncChunk,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Get sync chunk

**Parameters**

<i>afterUSN</i>	The USN after which the sync chunks are being requested
<i>maxEntries</i>	Max number of items within the sync chunk to be returned
<i>filter</i>	Filter for items to be returned within the sync chunks
<i>syncChunk</i>	Output parameter, the sync chunk
<i>errorDescription</i>	The textual description of the error in case sync chunk could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

**Returns**

Error code, 0 in case of successful sync chunk retrieval, other values corresponding to qevercloud::↵  
EDAMErrorCode enumeration instead

**4.31.2.13 getSyncState()**

```
virtual qint32 quentier::INoteStore::getSyncState (
    qevercloud::SyncState & syncState,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Get sync state

**Parameters**

<i>syncState</i>	Output parameter, the sync state
<i>errorDescription</i>	The textual description of the error in case sync state could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

**Returns**

Error code, 0 in case of successful sync state retrieval, other values corresponding to qevercloud::↵  
EDAMErrorCode enumeration instead

**4.31.2.14 noteStoreUrl()**

```
virtual QString quantier::INoteStore::noteStoreUrl ( ) const [pure virtual]
```

Provide note store URL used by this [INoteStore](#) instance

**4.31.2.15 setAuthData()**

```
virtual void quantier::INoteStore::setAuthData (
    QString authenticationToken,
    QList< QNetworkCookie > cookies ) [pure virtual]
```

Set authentication data to be used by this [INoteStore](#) instance

**4.31.2.16 setNoteStoreUrl()**

```
virtual void quantier::INoteStore::setNoteStoreUrl (
    QString noteStoreUrl ) [pure virtual]
```

Set note store URL to be used by this [INoteStore](#) instance

**4.31.2.17 stop()**

```
virtual void quantier::INoteStore::stop ( ) [pure virtual]
```

Stop asynchronous queries for notes or resources which might be running at the moment

**4.31.2.18 updateNote()**

```
virtual qint32 quantier::INoteStore::updateNote (
    Note & note,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds,
    QString linkedNotebookAuthToken = {} ) [pure virtual]
```

Update note

**Parameters**

<i>note</i>	<a href="#">Note</a> to be updated, must have guid set
<i>errorDescription</i>	The textual description of the error in case note could not be updated
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>
<i>linkedNotebookAuthToken</i>	If a note is updated within another user's account, the corresponding auth token should be set, otherwise the note would be updated within user's own account

**Returns**

Error code, 0 in case of successful note update, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

**4.31.2.19 updateNotebook()**

```
virtual qint32 qentier::INoteStore::updateNotebook (
    Notebook & notebook,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds,
    QString linkedNotebookAuthToken = {} ) [pure virtual]
```

Update notebook

**Parameters**

<i>notebook</i>	<a href="#">Notebook</a> to be updated, must have guid set
<i>errorDescription</i>	The textual description of the error in case notebook could not be updated
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>
<i>linkedNotebookAuthToken</i>	If a notebook is updated within another user's account, the corresponding auth token should be set, otherwise the notebook would be updated within user's own account

**Returns**

Error code, 0 in case of successful notebook update, other values corresponding to `qevercloud::EDAMError↵` Code enumeration instead

**4.31.2.20 updateSavedSearch()**

```
virtual qint32 qentier::INoteStore::updateSavedSearch (
    SavedSearch & savedSearch,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Update saved search

**Parameters**

<i>savedSearch</i>	Saved search to be updated, must have guid set
<i>errorDescription</i>	The textual description of the error in case saved search could not be updated
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>

## Returns

Error code, 0 in case of successful saved search update, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

## 4.31.2.21 updateTag()

```
virtual qint32 quantier::INoteStore::updateTag (
    Tag & tag,
    ErrorString & errorDescription,
    qint32 & rateLimitSeconds,
    QString linkedNotebookAuthToken = {} ) [pure virtual]
```

## Update tag

## Parameters

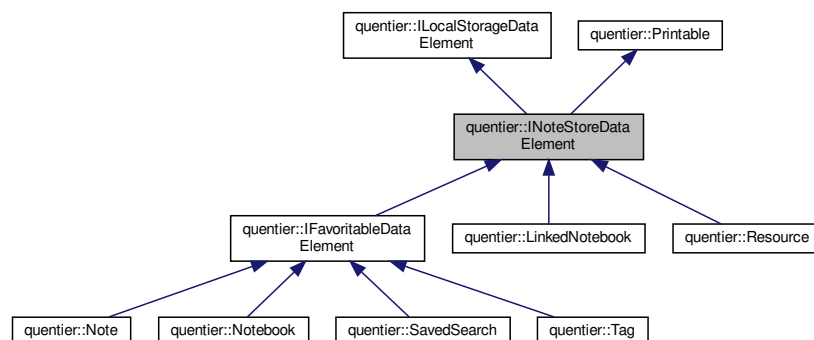
<i>tag</i>	<a href="#">Tag</a> to be updated, must have guid set
<i>errorDescription</i>	The textual description of the error in case tag could not be updated
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches <code>qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED</code>
<i>linkedNotebookAuthToken</i>	If a tag is updated within another user's account, the corresponding auth token should be set, otherwise the tag would be updated within user's own account

## Returns

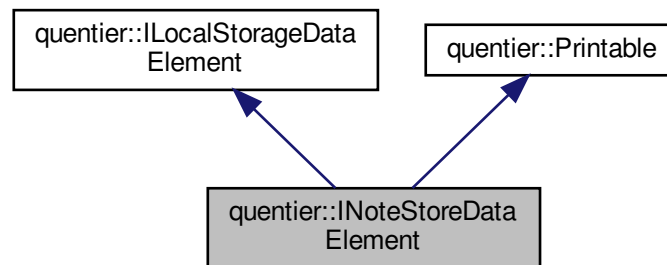
Error code, 0 in case of successful tag update, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

## 4.32 quantier::INoteStoreDataElement Class Reference

Inheritance diagram for `quantier::INoteStoreDataElement`:



Collaboration diagram for `quentier::INoteStoreDataElement`:



## Public Member Functions

- virtual void **clear** ()=0
- virtual bool **hasGuid** () const =0
- virtual const QString & **guid** () const =0
- virtual void **setGuid** (const QString &guid)=0
- virtual bool **hasUpdateSequenceNumber** () const =0
- virtual qint32 **updateSequenceNumber** () const =0
- virtual void **setUpdateSequenceNumber** (const qint32 usn)=0
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const =0
- virtual bool **isDirty** () const =0
- virtual void **setDirty** (const bool dirty)=0
- virtual bool **isLocal** () const =0
- virtual void **setLocal** (const bool local)=0

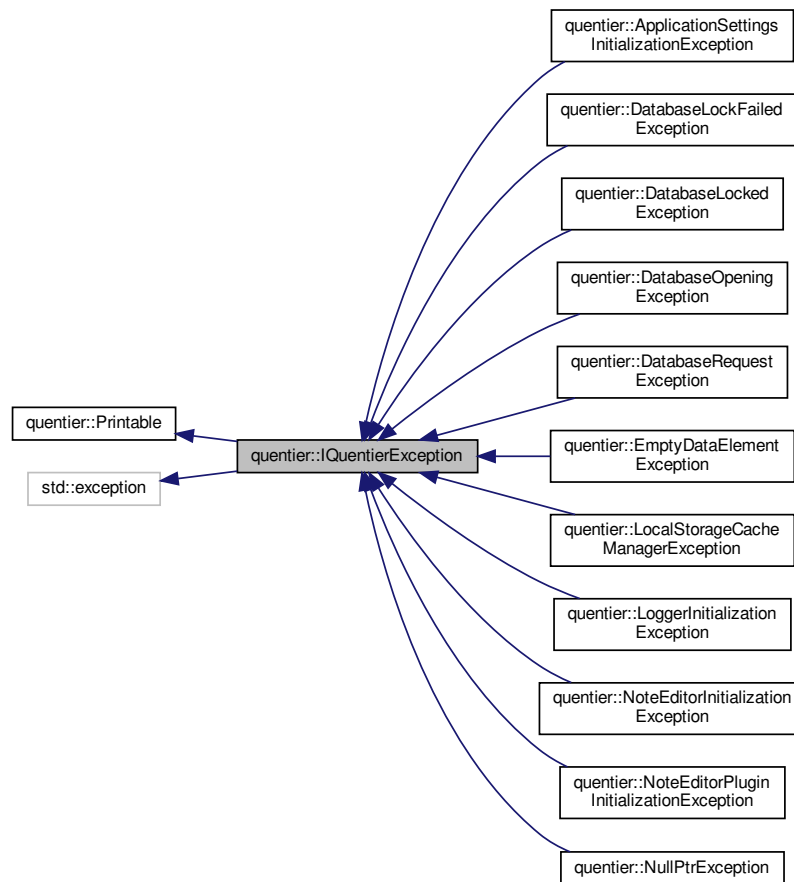
## Additional Inherited Members

## 4.33 quentier::IQuentierException Class Reference

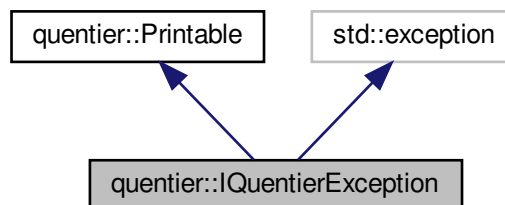
The [IQuentierException](#) class represents the interface for exceptions specific to libquentier and applications based on it.

```
#include <IQuentierException.h>
```

Inheritance diagram for quantier::IQuantierException:



Collaboration diagram for quantier::IQuantierException:



## Public Member Functions

- `IQuantierException` (const [ErrorString](#) &message)

- QString **localizedErrorMessage** () const
- QString **nonLocalizedErrorMessage** () const
- virtual const char \* **what** () const noexcept override
- virtual QTextStream & **print** (QTextStream &strm) const override

## Protected Member Functions

- **IQuentierException** (const IQuentierException &other)
- IQuentierException & **operator=** (const IQuentierException &other)
- virtual const QString **exceptionDisplayName** () const =0

### 4.33.1 Detailed Description

The [IQuentierException](#) class represents the interface for exceptions specific to libquentier and applications based on it.

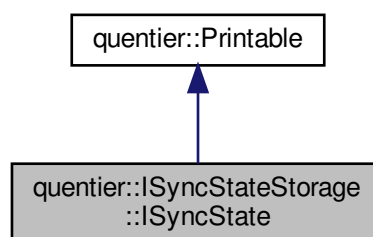
In addition to standard exception features inherited from std::exception, [IQuentierException](#) based exceptions can provide both localized and non-localized error messages.

## 4.34 quentier::ISyncStateStorage::ISyncState Class Reference

The [ISyncState](#) interface provides accessory methods to determine the sync state for the account.

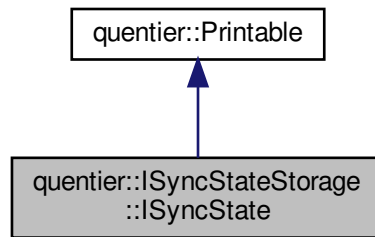
```
#include <ISyncStateStorage.h>
```

Inheritance diagram for quentier::ISyncStateStorage::ISyncState:





Collaboration diagram for quantier::ISyncStateStorage::ISyncState:



### Public Member Functions

- virtual qint32 **userDataUpdateCount** () const =0
- virtual qevercloud::Timestamp **userDataLastSyncTime** () const =0
- virtual QHash< QString, qint32 > **linkedNotebookUpdateCounts** () const =0
- virtual QHash< QString, qevercloud::Timestamp > **linkedNotebookLastSyncTimes** () const =0
- virtual QTextStream & **print** (QTextStream &strm) const override

### Additional Inherited Members

#### 4.34.1 Detailed Description

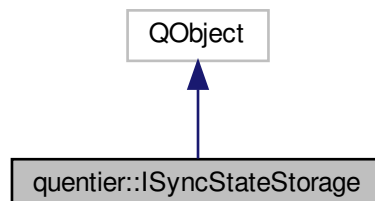
The [ISyncState](#) interface provides accessory methods to determine the sync state for the account.

## 4.35 quantier::ISyncStateStorage Class Reference

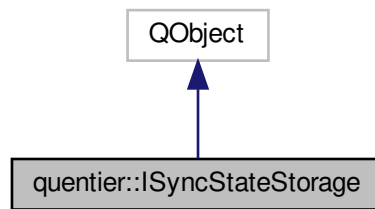
The [ISyncStateStorage](#) interface represents the interface of a class which stores sync state for given accounts persistently and provides access to previously stores sync states.

```
#include <ISyncStateStorage.h>
```

Inheritance diagram for quantier::ISyncStateStorage:



Collaboration diagram for `quentier::ISyncStateStorage`:



## Classes

- class [ISyncState](#)

*The [ISyncState](#) interface provides accessory methods to determine the sync state for the account.*

## Public Types

- using **ISyncStatePtr** = `std::shared_ptr< ISyncState >`

## Signals

- void [notifySyncStateUpdated](#) ([Account](#) account, [ISyncStatePtr](#) syncState)

## Public Member Functions

- **ISyncStateStorage** (`QObject *parent=nullptr`)
- virtual [ISyncStatePtr](#) **getSyncState** (`const Account &account`)=0
- virtual void **setSyncState** (`const Account &account, ISyncStatePtr syncState`)=0

### 4.35.1 Detailed Description

The [ISyncStateStorage](#) interface represents the interface of a class which stores sync state for given accounts persistently and provides access to previously stores sync states.

### 4.35.2 Member Function Documentation

#### 4.35.2.1 notifySyncStateUpdated

```
void quantier::ISyncStateStorage::notifySyncStateUpdated (
    Account account,
    ISyncStatePtr syncState ) [signal]
```

Classes implementing [ISyncStateStorage](#) interface are expected to emit notifySyncStateUpdated signal each time when sync state for the corresponding account is updated

## 4.36 quantier::IUserStore Class Reference

[IUserStore](#) is the interface which provides methods required for the implementation of UserStore part of Evernote EDAM sync protocol.

```
#include <IUserStore.h>
```

### Public Member Functions

- virtual void [setAuthData](#) (QString authenticationToken, QList< QNetworkCookie > cookies)=0
- virtual bool [checkVersion](#) (const QString &clientName, qint16 edamVersionMajor, qint16 edamVersionMinor, [ErrorString](#) &errorDescription)=0
- virtual qint32 [getUser](#) (User &user, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0
- virtual qint32 [getAccountLimits](#) (const qevercloud::ServiceLevel serviceLevel, qevercloud::AccountLimits &limits, [ErrorString](#) &errorDescription, qint32 &rateLimitSeconds)=0

#### 4.36.1 Detailed Description

[IUserStore](#) is the interface which provides methods required for the implementation of UserStore part of Evernote EDAM sync protocol.

#### 4.36.2 Member Function Documentation

##### 4.36.2.1 checkVersion()

```
virtual bool quantier::IUserStore::checkVersion (
    const QString & clientName,
    qint16 edamVersionMajor,
    qint16 edamVersionMinor,
    ErrorString & errorDescription ) [pure virtual]
```

Check the version of EDAM protocol

##### Parameters

<i>clientName</i>	Application name + application version + platform name string
<i>edamVersionMajor</i>	The major version of EDAM protocol the application wants to use to connect to Evernote
<i>edamVersionMinor</i>	The minor version of EDAM protocol the application wants to use to connect to Evernote
<i>errorDescription</i>	The textual description of the error if the supplied protocol version cannot be used to connect to Evernote

**Returns**

True if protocol check was successful i.e. the service can talk to the client using the supplied protocol version, false otherwise

**4.36.2.2 getAccountLimits()**

```
virtual qint32 quentier::IUserStore::getAccountLimits (
    const qevercloud::ServiceLevel serviceLevel,
    qevercloud::AccountLimits & limits,
    QString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Retrieve account limits corresponding to certain provided service level

**Parameters**

<i>serviceLevel</i>	The level of Evernote service for which account limits are requested
<i>limits</i>	Output account limits
<i>errorDescription</i>	The textual description of the error if account limits could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

**Returns**

Error code, 0 in case of successful retrieval of account limits for the given service level, other values corresponding to qevercloud::EDAMErrorCode::type enumeration instead

**4.36.2.3 getUser()**

```
virtual qint32 quentier::IUserStore::getUser (
    User & user,
    QString & errorDescription,
    qint32 & rateLimitSeconds ) [pure virtual]
```

Retrieve full information about user (account)

**Parameters**

<i>user</i>	Input and output parameter; on input needs to have user id set
<i>errorDescription</i>	The textual description of the error if full user information could not be retrieved
<i>rateLimitSeconds</i>	Output parameter, the number of seconds the client needs to wait before attempting to call this method or any other method calling Evernote API again; only meaningful if returned value matches qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED

**Returns**

Error code, 0 in case of successful retrieval of full user information, other values corresponding to `qevercloud::EDAMErrorCode` enumeration instead

**4.36.2.4 `setAuthData()`**

```
virtual void quentier::IUserStore::setAuthData (
    QString authenticationToken,
    QList< QNetworkCookie > cookies ) [pure virtual]
```

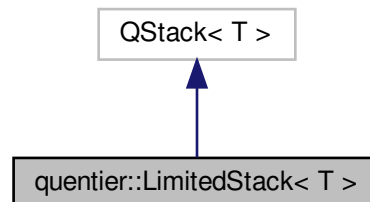
Set authentication data to be used by this [IUserStore](#) instance

**4.37 `quentier::LimitedStack< T >` Class Template Reference**

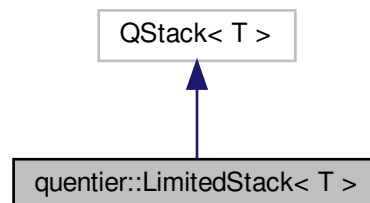
The [LimitedStack](#) template class implements a stack which may have a limitation for its size; when the size becomes too much according to the limit, the bottom element of the stack gets erased from it. Only limits greater than zero are considered.

```
#include <LimitedStack.h>
```

Inheritance diagram for `quentier::LimitedStack< T >`:



Collaboration diagram for `quentier::LimitedStack< T >`:



## Public Member Functions

- **LimitedStack** (void(\*deleter)(T &)=nullptr)
- **LimitedStack** (const [LimitedStack](#)< T > &other)
- **LimitedStack** ([LimitedStack](#)< T > &&other)
- [LimitedStack](#)< T > & **operator=** (const [LimitedStack](#)< T > &other)
- [LimitedStack](#)< T > & **operator=** ([LimitedStack](#)< T > &&other)
- void **swap** (const [LimitedStack](#)< T > &other)
- int **limit** () const
- void **setLimit** (const int limit)
- void **push** (const T &t)

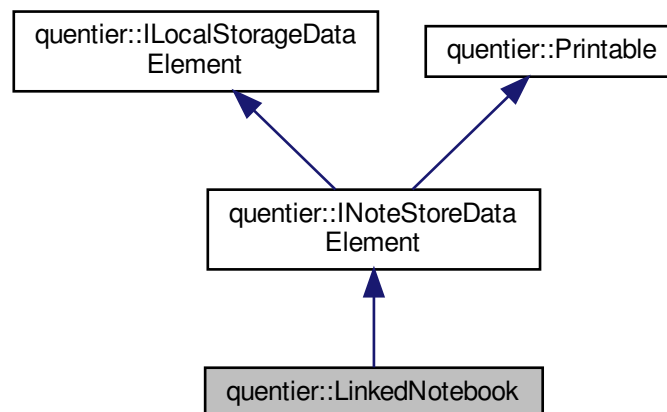
### 4.37.1 Detailed Description

```
template<class T>
class quantier::LimitedStack< T >
```

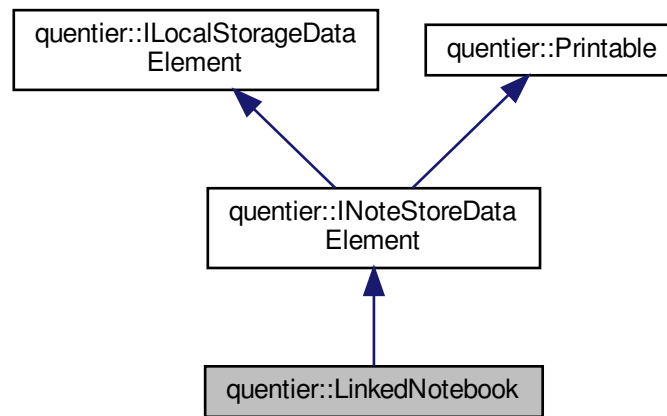
The [LimitedStack](#) template class implements a stack which may have a limitation for its size; when the size becomes too much according to the limit, the bottom element of the stack gets erased from it. Only limits greater than zero are considered.

## 4.38 quantier::LinkedNotebook Class Reference

Inheritance diagram for quantier::LinkedNotebook:



Collaboration diagram for quantier::LinkedNotebook:



## Public Member Functions

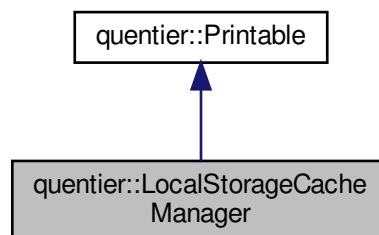
- **LinkedNotebook** (const [LinkedNotebook](#) &other)
- **LinkedNotebook** ([LinkedNotebook](#) &&other)
- [LinkedNotebook](#) & **operator=** (const [LinkedNotebook](#) &other)
- [LinkedNotebook](#) & **operator=** ([LinkedNotebook](#) &&other)
- **LinkedNotebook** (const qevercloud::LinkedNotebook &linkedNotebook)
- **LinkedNotebook** (qevercloud::LinkedNotebook &&linkedNotebook)
- const qevercloud::LinkedNotebook & **qevercloudLinkedNotebook** () const
- qevercloud::LinkedNotebook & **qevercloudLinkedNotebook** ()
- bool **operator==** (const [LinkedNotebook](#) &other) const
- bool **operator!=** (const [LinkedNotebook](#) &other) const
- virtual void **clear** () override
- virtual bool **hasGuid** () const override
- virtual const QString & **guid** () const override
- virtual void **setGuid** (const QString &guid) override
- virtual bool **hasUpdateSequenceNumber** () const override
- virtual quint32 **updateSequenceNumber** () const override
- virtual void **setUpdateSequenceNumber** (const quint32 usn) override
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const override
- bool **hasShareName** () const
- const QString & **shareName** () const
- void **setShareName** (const QString &shareName)
- bool **hasUsername** () const
- const QString & **username** () const
- void **setUsername** (const QString &username)
- bool **hasShardId** () const
- const QString & **shardId** () const
- void **setShardId** (const QString &shardId)
- bool **hasSharedNotebookGlobalId** () const
- const QString & **sharedNotebookGlobalId** () const

- void **setSharedNotebookGlobalId** (const QString &sharedNotebookGlobalId)
- bool **hasUri** () const
- const QString & **uri** () const
- void **setUri** (const QString &uri)
- bool **hasNoteStoreUrl** () const
- const QString & **noteStoreUrl** () const
- void **setNoteStoreUrl** (const QString &noteStoreUrl)
- bool **hasWebApiUrlPrefix** () const
- const QString & **webApiUrlPrefix** () const
- void **setWebApiUrlPrefix** (const QString &webApiUrlPrefix)
- bool **hasStack** () const
- const QString & **stack** () const
- void **setStack** (const QString &stack)
- bool **hasBusinessId** () const
- qint32 **businessId** () const
- void **setBusinessId** (const qint32 businessId)
- virtual QTextStream & **print** (QTextStream &strm) const override

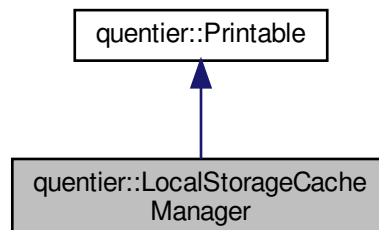
### Additional Inherited Members

## 4.39 quantier::LocalStorageCacheManager Class Reference

Inheritance diagram for quantier::LocalStorageCacheManager:



Collaboration diagram for quantier::LocalStorageCacheManager:





## Public Types

- enum **WhichUid** { **LocalUid** , **Guid** }

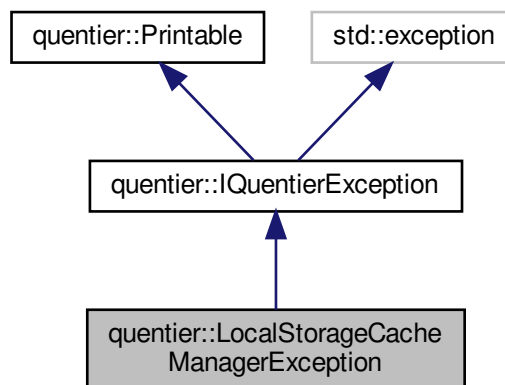
## Public Member Functions

- void **clear** ()
- bool **empty** () const
- size\_t **numCachedNotes** () const
- void **cacheNote** (const [Note](#) &note)
- void **expungeNote** (const [Note](#) &note)
- const [Note](#) \* **findNote** (const QString &uid, const WhichUid whichUid) const
- void **clearAllNotes** ()
- size\_t **numCachedResources** () const
- void **cacheResource** (const [Resource](#) &resource)
- void **expungeResource** (const [Resource](#) &resource)
- const [Resource](#) \* **findResource** (const QString &id, const WhichUid whichUid) const
- void **clearAllResources** ()
- size\_t **numCachedNotebooks** () const
- void **cacheNotebook** (const [Notebook](#) &notebook)
- void **expungeNotebook** (const [Notebook](#) &notebook)
- const [Notebook](#) \* **findNotebook** (const QString &uid, const WhichUid whichUid) const
- const [Notebook](#) \* **findNotebookByName** (const QString &name) const
- void **clearAllNotebooks** ()
- size\_t **numCachedTags** () const
- void **cacheTag** (const [Tag](#) &tag)
- void **expungeTag** (const [Tag](#) &tag)
- const [Tag](#) \* **findTag** (const QString &uid, const WhichUid whichUid) const
- const [Tag](#) \* **findTagByName** (const QString &name) const
- void **clearAllTags** ()
- size\_t **numCachedLinkedNotebooks** () const
- void **cacheLinkedNotebook** (const [LinkedNotebook](#) &linkedNotebook)
- void **expungeLinkedNotebook** (const [LinkedNotebook](#) &linkedNotebook)
- const [LinkedNotebook](#) \* **findLinkedNotebook** (const QString &guid) const
- void **clearAllLinkedNotebooks** ()
- size\_t **numCachedSavedSearches** () const
- void **cacheSavedSearch** (const [SavedSearch](#) &savedSearch)
- void **expungeSavedSearch** (const [SavedSearch](#) &savedSearch)
- const [SavedSearch](#) \* **findSavedSearch** (const QString &uid, const WhichUid whichUid) const
- const [SavedSearch](#) \* **findSavedSearchByName** (const QString &name) const
- void **clearAllSavedSearches** ()
- void **installCacheExpiryFunction** (const [ILocalStorageCacheExpiryChecker](#) &checker)
- virtual QTextStream & **print** (QTextStream &strm) const override

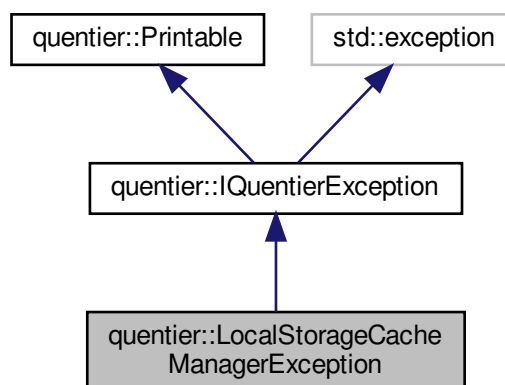
## Additional Inherited Members

### 4.40 `quentier::LocalStorageCacheManagerException` Class Reference

Inheritance diagram for `quentier::LocalStorageCacheManagerException`:



Collaboration diagram for `quentier::LocalStorageCacheManagerException`:



## Public Member Functions

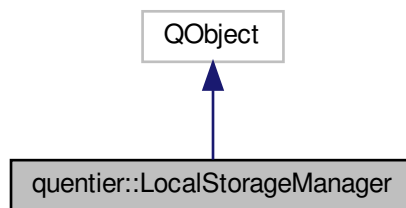
- `LocalStorageCacheManagerException` (const [ErrorString](#) &message)

## Protected Member Functions

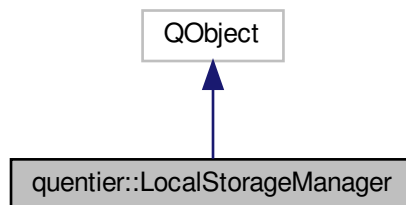
- virtual const QString **exceptionDisplayName** () const override

## 4.41 quantier::LocalStorageManager Class Reference

Inheritance diagram for quantier::LocalStorageManager:



Collaboration diagram for quantier::LocalStorageManager:



## Public Types

- enum class [StartupOption](#) { `ClearDatabase` = 1 , `OverrideLock` = 2 }

*The StartupOption enum is a QFlags enum which allows to specify some options to be applied to the local storage database on startup or on call to switchUser method.*

- enum class [ListObjectsOption](#) {  
**ListAll** = 0 , **ListDirty** = 1 , **ListNonDirty** = 2 , **ListElementsWithoutGuid** = 4 ,  
**ListElementsWithGuid** = 8 , **ListLocal** = 16 , **ListNonLocal** = 32 , **ListFavoritedElements** = 64 ,  
**ListNonFavoritedElements** = 128 }

*The ListObjectsOption enum is a QFlags enum which allows to specify the desired local storage elements in calls to methods listing them from the database.*

- enum class [OrderDirection](#) { **Ascending** = 0 , **Descending** }

The `OrderDirection` enum specifies the direction of ordering of the results for methods listing the objects from the local storage database.

- enum class `ListNotebooksOrder` {  
**ByUpdateSequenceNumber** = 0 , **ByNotebookName** , **ByCreationTimestamp** , **ByModificationTimestamp** ,  
**NoOrder** }

The `ListNotebooksOrder` allows to specify the results ordering for methods listing notebooks from the local storage database.

- enum class `ListLinkedNotebooksOrder` { **ByUpdateSequenceNumber** = 0 , **ByShareName** , **ByUsername** , **NoOrder** }

The `ListLinkedNotebooksOrder` enum allows to specify the results ordering for methods listing linked notebooks from local storage.

- enum class `NoteCountOption` { **IncludeNonDeletedNotes** = 1 , **IncludeDeletedNotes** = 2 }

The `NoteCountOption` enum is a `QFlags` enum which allows to specify some options for methods returning note counts from local storage.

- enum class `UpdateNoteOption` { **UpdateResourceMetadata** = 1 , **UpdateResourceBinaryData** = 2 , **UpdateTags** = 4 }

The `UpdateNoteOption` enum is a `QFlags` enum which allows to specify which note fields should be updated when `updateNote` method is called.

- enum class `GetNoteOption` { **WithResourceMetadata** = 1 , **WithResourceBinaryData** = 2 }

The `GetNoteOption` enum is a `QFlags` enum which allows to specify which note fields should be included when `findNote` or one of `listNote*` methods is called.

- enum class `ListNotesOrder` {  
**ByUpdateSequenceNumber** = 0 , **ByTitle** , **ByCreationTimestamp** , **ByModificationTimestamp** ,  
**ByDeletionTimestamp** , **ByAuthor** , **BySource** , **BySourceApplication** ,  
**ByReminderTime** , **ByPlaceName** , **NoOrder** }

The `ListNotesOrder` enum allows to specify the results ordering for methods listing notes from the local storage database.

- enum class `ListTagsOrder` { **ByUpdateSequenceNumber** , **ByName** , **NoOrder** }

The `ListTagsOrder` enum allows to specify the results ordering for methods listing tags from the local storage database.

- enum class `GetResourceOption` { **WithBinaryData** = 1 }

The `GetResourceOption` enum is a `QFlags` enum which allows to specify which resource fields should be included when `findEnResource` method is called.

- enum class `ListSavedSearchesOrder` { **ByUpdateSequenceNumber** = 0 , **ByName** , **ByFormat** , **NoOrder** }

The `ListSavedSearchesOrder` enum allows to specify the results ordering for methods listing saved searches from local storage.

## Signals

- void `upgradeProgress` (double progress)

`LocalStorageManager` is capable of performing automatic database upgrades if/when it is necessary.

## Public Member Functions

- `LocalStorageManager` (const `Account` &account, const `StartupOptions` options={}, `QObject` \*parent=nullptr)  
`LocalStorageManager` - constructor. Takes in the account for which the `LocalStorageManager` instance is created plus some other parameters determining the startup behaviour.
- void `switchUser` (const `Account` &account, const `StartupOptions` options={})  
`switchUser` - switches to another local storage database file associated with the passed in account
- bool `isLocalStorageVersionTooHigh` (`ErrorString` &errorDescription)
- bool `localStorageRequiresUpgrade` (`ErrorString` &errorDescription)
- `QVector`< std::shared\_ptr< `ILocalStoragePatch` > > `requiredLocalStoragePatches` ()

- qint32 [localStorageVersion](#) (ErrorString &errorDescription)
- qint32 [highestSupportedLocalStorageVersion](#) () const
- int [userCount](#) (ErrorString &errorDescription) const  
*userCount returns the number of non-deleted users currently stored in the local storage database*
- bool [addUser](#) (const User &user, ErrorString &errorDescription)  
*addUser adds the passed in User object to the local storage database*
- bool [updateUser](#) (const User &user, ErrorString &errorDescription)  
*updateUser updates the passed in User object in the local storage database*
- bool [findUser](#) (User &user, ErrorString &errorDescription) const  
*findUser attempts to find and fill the fields of the passed in User object which must have "id" field set as this value is used as the identifier of User objects in the local storage database*
- bool [deleteUser](#) (const User &user, ErrorString &errorDescription)  
*deleteUser marks the user as deleted in local storage*
- bool [expungeUser](#) (const User &user, ErrorString &errorDescription)  
*expungeUser permanently deletes the user from the local storage database*
- int [notebookCount](#) (ErrorString &errorDescription) const  
*notebookCount returns the number of notebooks currently stored in the local storage database*
- bool [addNotebook](#) (Notebook &notebook, ErrorString &errorDescription)  
*addNotebook adds the passed in Notebook to the local storage database*
- bool [updateNotebook](#) (Notebook &notebook, ErrorString &errorDescription)  
*updateNotebook updates the passed in Notebook in the local storage database*
- bool [findNotebook](#) (Notebook &notebook, ErrorString &errorDescription) const  
*findNotebook attempts to find and set all found fields of the passed in Notebook object*
- bool [findDefaultNotebook](#) (Notebook &notebook, ErrorString &errorDescription) const  
*findDefaultNotebook attempts to find the default notebook in the local storage database.*
- bool [findLastUsedNotebook](#) (Notebook &notebook, ErrorString &errorDescription) const  
*findLastUsedNotebook attempts to find the last used notebook in the local storage database.*
- bool [findDefaultOrLastUsedNotebook](#) (Notebook &notebook, ErrorString &errorDescription) const  
*findDefaultOrLastUsedNotebook attempts to find either the default or the last used notebook in the local storage database.*
- QList< Notebook > [listAllNotebooks](#) (ErrorString &errorDescription, const size\_t limit=0, const size\_t offset=0, const ListNotebooksOrder order=ListNotebooksOrder::NoOrder, const OrderDirection orderDirection=OrderDirection::Ascending, const QString &linkedNotebookGuid=QString()) const  
*listAllNotebooks attempts to list all notebooks within the current account from the local storage database.*
- QList< Notebook > [listNotebooks](#) (const ListObjectsOptions flag, ErrorString &errorDescription, const size\_t limit=0, const size\_t offset=0, const ListNotebooksOrder order=ListNotebooksOrder::NoOrder, const OrderDirection orderDirection=OrderDirection::Ascending, const QString &linkedNotebookGuid=QString()) const  
*listNotebooks attempts to list notebooks within the account according to the specified input flag*
- QList< SharedNotebook > [listAllSharedNotebooks](#) (ErrorString &errorDescription) const  
*listAllSharedNotebooks attempts to list all shared notebooks within the account.*
- QList< SharedNotebook > [listSharedNotebooksPerNotebookGuid](#) (const QString &notebookGuid, ErrorString &errorDescription) const  
*listSharedNotebooksPerNotebookGuid - attempts to list all shared notebooks per given notebook's remote guid (not local uid, it's important).*
- bool [expungeNotebook](#) (Notebook &notebook, ErrorString &errorDescription)  
*expungeNotebook permanently deletes the specified notebook from the local storage database.*
- int [linkedNotebookCount](#) (ErrorString &errorDescription) const  
*linkedNotebookCount returns the number of linked notebooks stored in the local storage database.*
- bool [addLinkedNotebook](#) (const LinkedNotebook &linkedNotebook, ErrorString &errorDescription)  
*addLinkedNotebook adds passed in LinkedNotebook to the local storage database; LinkedNotebook must have "remote" Evernote service's guid set. It is not possible to add a linked notebook in offline mode so it doesn't make sense for LinkedNotebook objects to not have guid.*

- bool `updateLinkedNotebook` (const `LinkedNotebook` &linkedNotebook, `ErrorString` &errorDescription)
 

*updateLinkedNotebook updates passwd in `LinkedNotebook` in the local storage database; `LinkedNotebook` must have "remote" Evernote service's guid set.*
- bool `findLinkedNotebook` (`LinkedNotebook` &linkedNotebook, `ErrorString` &errorDescription) const
 

*findLinkedNotebook attempts to find and set all found fields for passed in by reference `LinkedNotebook` object. For `LinkedNotebook` local uid doesn't mean anything because it can only be considered valid if it has "remote" Evernote service's guid set. So this passed in `LinkedNotebook` object must have guid set to identify the linked notebook in the local storage database.*
- `QList< LinkedNotebook > listAllLinkedNotebooks` (`ErrorString` &errorDescription, const size\_t limit=0, const size\_t offset=0, const `ListLinkedNotebooksOrder` order=ListLinkedNotebooksOrder::NoOrder, const `OrderDirection` orderDirection=OrderDirection::Ascending) const
 

*listAllLinkedNotebooks - attempts to list all linked notebooks within the account.*
- `QList< LinkedNotebook > listLinkedNotebooks` (const `ListObjectsOptions` flag, `ErrorString` &errorDescription, const size\_t limit=0, const size\_t offset=0, const `ListLinkedNotebooksOrder` order=ListLinkedNotebooksOrder::NoOrder, const `OrderDirection` orderDirection=OrderDirection::Ascending) const
 

*listLinkedNotebooks attempts to list linked notebooks within the account according to the specified input flag.*
- bool `expungeLinkedNotebook` (const `LinkedNotebook` &linkedNotebook, `ErrorString` &errorDescription)
 

*expungeLinkedNotebook permanently deletes specified linked notebook from the local storage database.*
- int `noteCount` (`ErrorString` &errorDescription, const `NoteCountOptions` options=NoteCountOption::IncludeNonDeletedNotes) const
 

*noteCount returns the number of notes currently stored in the local storage database.*
- int `noteCountPerNotebook` (const `Notebook` &notebook, `ErrorString` &errorDescription, const `NoteCountOptions` options=NoteCountOption::IncludeNonDeletedNotes) const
 

*noteCountPerNotebook returns the number of notes currently stored in the local storage database per given notebook.*
- int `noteCountPerTag` (const `Tag` &tag, `ErrorString` &errorDescription, const `NoteCountOptions` options=NoteCountOption::IncludeNonDeletedNotes) const
 

*noteCountPerTag returns the number of notes currently stored in local storage database labeled with given tag.*
- bool `noteCountsPerAllTags` (`QHash< QString, int >` &noteCountsPerTagLocalUids, `ErrorString` &errorDescription, const `NoteCountOptions` options=NoteCountOption::IncludeNonDeletedNotes) const
 

*noteCountsPerAllTags returns the number of notes currently stored in local storage database labeled with each tag stored in the local storage database.*
- int `noteCountPerNotebooksAndTags` (const `QStringList` &notebookLocalUids, const `QStringList` &tagLocalUids, `ErrorString` &errorDescription, const `NoteCountOptions` options=NoteCountOption::IncludeNonDeletedNotes) const
 

*noteCountPerNotebooksAndTags returns the number of notes currently stored in local storage database belonging to one of notebooks corresponding to given notebook local uids and labeled by at least one of tags corresponding to given tag local uids*
- bool `addNote` (`Note` &note, `ErrorString` &errorDescription)
 

*addNote adds passed in `Note` to the local storage database.*
- bool `updateNote` (`Note` &note, const `UpdateNoteOptions` options, `ErrorString` &errorDescription)
 

*updateNote updates passed in `Note` in the local storage database.*
- bool `findNote` (`Note` &note, const `GetNoteOptions` options, `ErrorString` &errorDescription) const
 

*findNote - attempts to find note in the local storage database*
- `QList< Note > listNotesPerNotebook` (const `Notebook` &notebook, const `GetNoteOptions` options, `ErrorString` &errorDescription, const `ListObjectsOptions` &flag=ListObjectsOption::ListAll, const size\_t limit=0, const size\_t offset=0, const `ListNotesOrder` &order=ListNotesOrder::NoOrder, const `OrderDirection` &orderDirection=OrderDirection::Ascending) const
 

*listNotesPerNotebook attempts to list notes per given notebook*
- `QList< Note > listNotesPerTag` (const `Tag` &tag, const `GetNoteOptions` options, `ErrorString` &errorDescription, const `ListObjectsOptions` &flag=ListObjectsOption::ListAll, const size\_t limit=0, const size\_t offset=0, const `ListNotesOrder` &order=ListNotesOrder::NoOrder, const `OrderDirection` &orderDirection=OrderDirection::Ascending) const
 

*listNotesPerTag attempts to list notes labeled with a given tag*

- `QList< Note > listNotesPerNotebooksAndTags` (const QStringList &notebookLocalUids, const QStringList &tagLocalUids, const GetNoteOptions options, [ErrorString](#) &errorDescription, const ListObjectsOptions &flag=ListObjectsOption::ListAll, const size\_t limit=0, const size\_t offset=0, const [ListNotesOrder](#) &order=ListNotesOrder::NoOrder, const [OrderDirection](#) &orderDirection=OrderDirection::Ascending) const  
*listNotesPerNotebooksAndTags attempts to list notes which are present within one of specified notebooks and are labeled with at least one of specified tags*
- `QList< Note > listNotesByLocalUids` (const QStringList &noteLocalUids, const GetNoteOptions options, [ErrorString](#) &errorDescription, const ListObjectsOptions &flag=ListObjectsOption::ListAll, const size\_t limit=0, const size\_t offset=0, const [ListNotesOrder](#) &order=ListNotesOrder::NoOrder, const [OrderDirection](#) &orderDirection=OrderDirection::Ascending) const  
*listNotesByLocalUids attempts to list notes given their local uids*
- `QList< Note > listNotes` (const ListObjectsOptions flag, const GetNoteOptions options, [ErrorString](#) &errorDescription, const size\_t limit=0, const size\_t offset=0, const [ListNotesOrder](#) order=ListNotesOrder::NoOrder, const [OrderDirection](#) orderDirection=OrderDirection::Ascending, const QString &linkedNotebookGuid=QString()) const  
*listNotes attempts to list notes within the account according to the specified input flag.*
- `QStringList findNoteLocalUidsWithSearchQuery` (const [NoteSearchQuery](#) &noteSearchQuery, [ErrorString](#) &errorDescription) const  
*findNoteLocalUidsWithSearchQuery attempts to find note local uids of notes corresponding to the passed in [NoteSearchQuery](#) object.*
- `NoteList findNotesWithSearchQuery` (const [NoteSearchQuery](#) &noteSearchQuery, const GetNoteOptions options, [ErrorString](#) &errorDescription) const  
*findNotesWithSearchQuery attempts to find notes corresponding to the passed in [NoteSearchQuery](#) object.*
- `bool expungeNote` ([Note](#) &note, [ErrorString](#) &errorDescription)  
*expungeNote permanently deletes note from local storage.*
- `int tagCount` ([ErrorString](#) &errorDescription) const  
*tagCount returns the number of non-deleted tags currently stored in the local storage database.*
- `bool addTag` ([Tag](#) &tag, [ErrorString](#) &errorDescription)  
*addTag adds passed in [Tag](#) to the local storage database. If tag has "remote" Evernote service's guid set, it is identified in the database by this guid. Otherwise it is identified by local uid.*
- `bool updateTag` ([Tag](#) &tag, [ErrorString](#) &errorDescription)  
*updateTag updates passed in [Tag](#) in the local storage database.*
- `bool findTag` ([Tag](#) &tag, [ErrorString](#) &errorDescription) const  
*findTag attempts to find and fill the fields of passed in tag object.*
- `QList< Tag > listAllTagsPerNote` (const [Note](#) &note, [ErrorString](#) &errorDescription, const ListObjectsOptions &flag=ListObjectsOption::ListAll, const size\_t limit=0, const size\_t offset=0, const [ListTagsOrder](#) &order=ListTagsOrder::NoOrder, const [OrderDirection](#) &orderDirection=OrderDirection::Ascending) const  
*listAllTagsPerNote lists all tags per given note*
- `QList< Tag > listAllTags` ([ErrorString](#) &errorDescription, const size\_t limit=0, const size\_t offset=0, const [ListTagsOrder](#) order=ListTagsOrder::NoOrder, const [OrderDirection](#) orderDirection=OrderDirection::Ascending, const QString &linkedNotebookGuid=QString()) const  
*listAllTags lists all tags within the current user's account.*
- `QList< Tag > listTags` (const ListObjectsOptions flag, [ErrorString](#) &errorDescription, const size\_t limit=0, const size\_t offset=0, const [ListTagsOrder](#) &order=ListTagsOrder::NoOrder, const [OrderDirection](#) orderDirection=OrderDirection::Ascending, const QString &linkedNotebookGuid=QString()) const  
*listTags attempts to list tags within the account according to the specified input flag.*
- `QList< std::pair< Tag, QStringList > > listTagsWithNoteLocalUids` (const ListObjectsOptions flag, [ErrorString](#) &errorDescription, const size\_t limit=0, const size\_t offset=0, const [ListTagsOrder](#) &order=ListTagsOrder::NoOrder, const [OrderDirection](#) orderDirection=OrderDirection::Ascending, const QString &linkedNotebookGuid=QString()) const  
*listTagsWithNoteLocalUids attempts to list tags and their corresponding local uids within the account according to the specified input flag*
- `bool expungeTag` ([Tag](#) &tag, QStringList &expungedChildTagLocalUids, [ErrorString](#) &errorDescription)  
*expungeTag permanently deletes tag from the local storage database.*



- bool [expungeNotelessTagsFromLinkedNotebooks](#) ([ErrorString](#) &errorDescription)  
*expungeNotelessTagsFromLinkedNotebooks permanently deletes from the local storage database those tags which belong to some linked notebook and are not linked with any notes.*
- int [enResourceCount](#) ([ErrorString](#) &errorDescription) const  
*enResourceCount (the name is not [Resource](#) to prevent problems with macro defined on some versions of Windows) returns the number of resources currently stored in the local storage database.*
- bool [addEnResource](#) ([Resource](#) &resource, [ErrorString](#) &errorDescription)  
*addEnResource adds passed in resource to the local storage database.*
- bool [updateEnResource](#) ([Resource](#) &resource, [ErrorString](#) &errorDescription)  
*updateEnResource updates passed in resource in the local storage database.*
- bool [findEnResource](#) ([Resource](#) &resource, const [GetResourceOptions](#) options, [ErrorString](#) &errorDescription) const  
*findEnResource method attempts to find resource in the local storage database*
- bool [expungeEnResource](#) ([Resource](#) &resource, [ErrorString](#) &errorDescription)  
*expungeResource permanently deletes resource from the local storage database.*
- int [savedSearchCount](#) ([ErrorString](#) &errorDescription) const  
*savedSearchCount returns the number of saved seacrhs currently stored in local storage database.*
- bool [addSavedSearch](#) ([SavedSearch](#) &search, [ErrorString](#) &errorDescription)  
*addSavedSearch adds passed in [SavedSearch](#) to the local storage database; if search has "remote" Evernote service's guid set, it is identified in the database by this guid. Otherwise it is identified by local uid.*
- bool [updateSavedSearch](#) ([SavedSearch](#) &search, [ErrorString](#) &errorDescription)  
*updateSavedSearch updates passed in [SavedSearch](#) in the local storage database.*
- bool [findSavedSearch](#) ([SavedSearch](#) &search, [ErrorString](#) &errorDescription) const  
*findSavedSearch attempts to find and fill the fields of passed in saved search object.*
- [QList](#)< [SavedSearch](#) > [listAllSavedSearches](#) ([ErrorString](#) &errorDescription, const [size\\_t](#) limit=0, const [size\\_t](#) offset=0, const [ListSavedSearchesOrder](#) order=[ListSavedSearchesOrder](#)::NoOrder, const [OrderDirection](#) orderDirection=[OrderDirection](#)::Ascending) const  
*listAllSavedSearches lists all saved searches within the account.*
- [QList](#)< [SavedSearch](#) > [listSavedSearches](#) (const [ListObjectsOptions](#) flag, [ErrorString](#) &errorDescription, const [size\\_t](#) limit=0, const [size\\_t](#) offset=0, const [ListSavedSearchesOrder](#) order=[ListSavedSearchesOrder](#)::NoOrder, const [OrderDirection](#) orderDirection=[OrderDirection](#)::Ascending) const  
*listSavedSearches attempts to list saved searches within the account according to the specified input flag.*
- bool [expungeSavedSearch](#) ([SavedSearch](#) &search, [ErrorString](#) &errorDescription)  
*expungeSavedSearch permanently deletes saved search from the local storage database.*
- [qint32](#) [accountHighUsn](#) (const [QString](#) &linkedNotebookGuid, [ErrorString](#) &errorDescription)  
*accountHighUsn returns the highest update sequence number within the data elements stored in the local storage database, either for user's own account or for some linked notebook.*

## Friends

- [QUENTIER\\_EXPORT](#) [QTextStream](#) & **operator**<< ([QTextStream](#) &strm, const [StartupOption](#) option)
- [QUENTIER\\_EXPORT](#) [QDebug](#) & **operator**<< ([QDebug](#) &dbg, const [StartupOption](#) option)
- [QUENTIER\\_EXPORT](#) [QTextStream](#) & **operator**<< ([QTextStream](#) &strm, const [StartupOptions](#) options)
- [QUENTIER\\_EXPORT](#) [QDebug](#) & **operator**<< ([QDebug](#) &dbg, const [StartupOptions](#) options)
- [QUENTIER\\_EXPORT](#) [QTextStream](#) & **operator**<< ([QTextStream](#) &strm, const [ListObjectsOption](#) option)
- [QUENTIER\\_EXPORT](#) [QDebug](#) & **operator**<< ([QDebug](#) &dbg, const [ListObjectsOption](#) option)
- [QUENTIER\\_EXPORT](#) [QTextStream](#) & **operator**<< ([QTextStream](#) &strm, const [ListObjectsOptions](#) options)
- [QUENTIER\\_EXPORT](#) [QDebug](#) & **operator**<< ([QDebug](#) &dbg, const [ListObjectsOptions](#) options)
- [QUENTIER\\_EXPORT](#) [QTextStream](#) & **operator**<< ([QTextStream](#) &strm, const [OrderDirection](#) orderDirection)
- [QUENTIER\\_EXPORT](#) [QDebug](#) & **operator**<< ([QDebug](#) &dbg, const [OrderDirection](#) orderDirection)
- [QUENTIER\\_EXPORT](#) [QTextStream](#) & **operator**<< ([QTextStream](#) &strm, const [ListNotebooksOrder](#) order)



- `QUENTIER_EXPORT QDebug & operator<< (QDebug &dbg, const ListNotebooksOrder order)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const ListLinkedNotebooksOrder order)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const ListLinkedNotebooksOrder order)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const NoteCountOption option)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &dbg, const NoteCountOption option)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const NoteCountOptions options)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const NoteCountOptions options)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const UpdateNoteOption option)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const UpdateNoteOption option)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const UpdateNoteOptions options)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const UpdateNoteOptions options)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const GetNoteOption option)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &dbg, const GetNoteOption option)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const GetNoteOptions options)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const GetNoteOptions options)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const ListNotesOrder order)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const ListNotesOrder order)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const ListTagsOrder order)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const ListTagsOrder order)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const GetResourceOption option)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const GetResourceOption option)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const GetResourceOptions options)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const GetResourceOptions options)`
- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const ListSavedSearchesOrder order)`
- `QUENTIER_EXPORT QDebug & operator<< (QDebug &strm, const ListSavedSearchesOrder order)`

## 4.41.1 Member Enumeration Documentation

### 4.41.1.1 `GetNoteOption`

```
enum quentier::LocalStorageManager::GetNoteOption [strong]
```

The `GetNoteOption` enum is a `QFlags` enum which allows to specify which note fields should be included when `findNote` or one of `listNote*` methods is called.

Most note data is included unconditionally - note title, content, attributes (if any) etc. However, some specific data can be opted to not be included into the returned note data - notably, metadata of resources and binary data of resources. If these are omitted, `findNote` or any of `listNote*` methods might work faster than otherwise

#### Enumerator

<code>WithResourceMetadata</code>	<code>WithResourceMetadata</code> value specifies that fields aside <code>dataBody</code> , <code>dataSize</code> , <code>dataHash</code> , <code>alternateDataBody</code> , <code>alternateDataSize</code> , <code>alternateDataHash</code> for each note's resource should be included
<code>WithResourceBinaryData</code>	<code>WithResourceBinaryData</code> value specifies that <code>dataBody</code> , its size and hash and <code>alternateDataBody</code> , its size and hash should be included into each of note's resources; this value only has effect if flags also have <code>WithResourceMetadata</code> value enabled!

#### 4.41.1.2 GetResourceOption

```
enum quentier::LocalStorageManager::GetResourceOption [strong]
```

The GetResourceOption enum is a QFlags enum which allows to specify which resource fields should be included when findEnResource method is called.

Most resource data is included unconditionally but some specific data can be opted to not be included into the returned resource data - notably, binary data of the resource. If it is omitted, findEnResource method might work faster than otherwise

##### Enumerator

WithBinaryData	WithBinaryData value specifies that dataBody and alternateDataBody should be included into the returned resource
----------------	--

#### 4.41.1.3 ListObjectsOption

```
enum quentier::LocalStorageManager::ListObjectsOption [strong]
```

The ListObjectsOption enum is a QFlags enum which allows to specify the desired local storage elements in calls to methods listing them from the database.

For example, one can either list all available elements of certain type from local storage or only elements marked as dirty (modified locally, not yet synchronized) or elements never synchronized with the remote storage or elements which are synchronizable with the remote storage etc.

#### 4.41.1.4 StartupOption

```
enum quentier::LocalStorageManager::StartupOption [strong]
```

The StartupOption enum is a QFlags enum which allows to specify some options to be applied to the local storage database on startup or on call to switchUser method.

##### Enumerator

ClearDatabase	If ClearDatabase flag is active, <a href="#">LocalStorageManager</a> would wipe any existing database contents; the net effect would be as if no database existed for the given user before the creation of <a href="#">LocalStorageManager</a> or before the call to its switchUser method
OverrideLock	If OverrideLock flag is active, <a href="#">LocalStorageManager</a> would ignore the existing advisory lock (if any) put on the database file; if this flag is not active, the attempt to create <a href="#">LocalStorageManager</a> (or the attempt to call its switchUser method) with the advisory lock on the database file put by someone else would cause the throwing of <a href="#">DatabaseLockedException</a>

#### 4.41.1.5 UpdateNoteOption

```
enum quentier::LocalStorageManager::UpdateNoteOption [strong]
```

The UpdateNoteOption enum is a QFlags enum which allows to specify which note fields should be updated when updateNote method is called.

Most note data is updated unconditionally - note title, content, attributes (if any) etc. However, some specific data can be chosen to not update - notably, metadata of resources, binary data of resources or lists of note's tags

##### Enumerator

UpdateResourceMetadata	UpdateResourceMetadata value specifies that fields aside dataBody, dataSize, dataHash, alternateDataBody, alternateDataSize, alternateDataHash for each note's resource should be updated
UpdateResourceBinaryData	UpdateResourceBinaryData value specifies that dataBody, its size and hash and alternateDataBody, its size and hash should be updated for each of note's resources; this value only has effect if flags also have UpdateResourceMetadata value enabled!
UpdateTags	UpdateTags value specifies that note's tag lists should be updated

### 4.41.2 Constructor & Destructor Documentation

#### 4.41.2.1 LocalStorageManager()

```
quentier::LocalStorageManager::LocalStorageManager (
    const Account & account,
    const StartupOptions options = {},
    QObject * parent = nullptr ) [explicit]
```

[LocalStorageManager](#) - constructor. Takes in the account for which the [LocalStorageManager](#) instance is created plus some other parameters determining the startup behaviour.

##### Parameters

<i>account</i>	The account for which the local storage is being created and initialized
<i>options</i>	Startup options for the local storage, none enabled by default
<i>parent</i>	Parent QObject

### 4.41.3 Member Function Documentation

#### 4.41.3.1 accountHighUsn()

```
qint32 quentier::LocalStorageManager::accountHighUsn (
    const QString & linkedNotebookGuid,
    ErrorString & errorDescription )
```

accountHighUsn returns the highest update sequence number within the data elements stored in the local storage database, either for user's own account or for some linked notebook.

##### Parameters

<i>linkedNotebookGuid</i>	The guid of the linked notebook for which the highest update sequence number is requested; if null or empty, the highest update sequence number for user's own account is returned
<i>errorDescription</i>	Error description if account's highest update sequence number could not be returned

##### Returns

Either the highest update sequence number - a non-negative value - or a negative number in case of error

#### 4.41.3.2 addEnResource()

```
bool quentier::LocalStorageManager::addEnResource (
    Resource & resource,
    ErrorString & errorDescription )
```

addEnResource adds passed in resource to the local storage database.

##### Parameters

<i>resource</i>	<a href="#">Resource</a> to be added to the database, must have either note's local uid set or note's "remote" Evernote service's guid set; may be changed as a result of the call, filled with autogenerated fields like local uid if it was empty before the call
<i>errorDescription</i>	Error description if resource could not be added

##### Returns

True if resource was added successfully, false otherwise

#### 4.41.3.3 addLinkedNotebook()

```
bool quentier::LocalStorageManager::addLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    ErrorString & errorDescription )
```

addLinkedNotebook adds passed in [LinkedNotebook](#) to the local storage database; [LinkedNotebook](#) must have "remote" Evernote service's guid set. It is not possible to add a linked notebook in offline mode so it doesn't make sense for [LinkedNotebook](#) objects to not have guid.

## Parameters

<i>linkedNotebook</i>	<a href="#">LinkedNotebook</a> to be added to the local storage database
<i>errorDescription</i>	Error description if linked notebook could not be added

## Returns

True if linked notebook was added successfully, false otherwise

#### 4.41.3.4 addNote()

```
bool quantier::LocalStorageManager::addNote (
    Note & note,
    ErrorString & errorDescription )
```

addNote adds passed in [Note](#) to the local storage database.

## Parameters

<i>note</i>	<a href="#">Note</a> to be added to local storage database; required to contain either "remote" notebook guid or local notebook uid; may be changed as a result of the call, filled with autogenerated fields like local uid if it was empty before the call; also tag guids are filled if the note passed in contained only tag local uids and tag local uids are filled if the note passed in contained only tag guids
<i>errorDescription</i>	Error description if note could not be added

## Returns

True if note was added successfully, false otherwise

#### 4.41.3.5 addNotebook()

```
bool quantier::LocalStorageManager::addNotebook (
    Notebook & notebook,
    ErrorString & errorDescription )
```

addNotebook adds the passed in [Notebook](#) to the local storage database

If the notebook has "remote" Evernote service's guid set, it is identified by this guid in the local storage database. Otherwise it is identified by the local uid

## Parameters

<i>notebook</i>	The notebook to be added to the local storage database; the object is passed by reference and may be changed as a result of the call (filled with autocompleted fields like local uid if it was empty before the call)
<i>errorDescription</i>	Error description if the notebook could not be added

**Returns**

True if the notebook was added successfully, false otherwise

**4.41.3.6 addSavedSearch()**

```
bool quentier::LocalStorageManager::addSavedSearch (
    SavedSearch & search,
    ErrorString & errorDescription )
```

addSavedSearch adds passed in [SavedSearch](#) to the local storage database; if search has "remote" Evernote service's guid set, it is identified in the database by this guid. Otherwise it is identified by local uid.

**Parameters**

<i>search</i>	<a href="#">SavedSearch</a> to be added to the local storage; may be changed as a result of the call, filled with autogenerated fields like local uid if it was empty before the call
<i>errorDescription</i>	Error description if <a href="#">SavedSearch</a> could not be added

**Returns**

True if [SavedSearch](#) was added successfully, false otherwise

**4.41.3.7 addTag()**

```
bool quentier::LocalStorageManager::addTag (
    Tag & tag,
    ErrorString & errorDescription )
```

addTag adds passed in [Tag](#) to the local storage database. If tag has "remote" Evernote service's guid set, it is identified in the database by this guid. Otherwise it is identified by local uid.

**Parameters**

<i>tag</i>	<a href="#">Tag</a> to be added to the local storage; may be changed as a result of the call, filled with autogenerated fields like local uid if it was empty before the call
<i>errorDescription</i>	Error description if <a href="#">Tag</a> could not be added

**Returns**

True if [Tag](#) was added successfully, false otherwise

#### 4.41.3.8 addUser()

```
bool quantier::LocalStorageManager::addUser (
    const User & user,
    ErrorString & errorDescription )
```

addUser adds the passed in [User](#) object to the local storage database

The table with Users is only involved in operations with notebooks which have "contact" field set which in turn is used with business accounts

##### Parameters

<i>user</i>	The user to be added to the local storage database
<i>errorDescription</i>	Error description if the user could not be added

##### Returns

True if the user was added successfully, false otherwise

#### 4.41.3.9 deleteUser()

```
bool quantier::LocalStorageManager::deleteUser (
    const User & user,
    ErrorString & errorDescription )
```

deleteUser marks the user as deleted in local storage

##### Parameters

<i>user</i>	The user to be marked as deleted
<i>errorDescription</i>	Error description if the user could not be marked as deleted

##### Returns

True if the user was marked as deleted successfully, false otherwise

#### 4.41.3.10 enResourceCount()

```
int quantier::LocalStorageManager::enResourceCount (
    ErrorString & errorDescription ) const
```

enResourceCount (the name is not [Resource](#) to prevent problems with macro defined on some versions of Windows) returns the number of resources currently stored in the local storage database.

**Parameters**

<i>errorDescription</i>	Error description if the number of resources could not be returned
-------------------------	--

**Returns**

Either non-negative value with the number of resources or -1 which means some error occurred

**4.41.3.11 expungeEnResource()**

```
bool quantier::LocalStorageManager::expungeEnResource (
    Resource & resource,
    ErrorString & errorDescription )
```

expungeResource permanently deletes resource from the local storage database.

**Parameters**

<i>resource</i>	Resource to be expunged; may be changed as a result of the call, automatically filled with local uid and note local uid and/or guid if these were empty before the call
<i>errorDescription</i>	Error description if resource could not be expunged

**Returns**

True if resource was expunged successfully, false otherwise

**4.41.3.12 expungeLinkedNotebook()**

```
bool quantier::LocalStorageManager::expungeLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    ErrorString & errorDescription )
```

expungeLinkedNotebook permanently deletes specified linked notebook from the local storage database.

Evernote API doesn't allow to delete linked notebooks from the remote storage, it can only be done by official desktop client or web client. So this method should be called only during the synchronization with remote service, when some linked notebook is found to be deleted via either official desktop client or web client.

**Parameters**

<i>linkedNotebook</i>	Linked notebook to be expunged. Must have "remote" guid set
<i>errorDescription</i>	Error description if linked notebook could not be expunged



#### Returns

True if linked notebook was expunged successfully, false otherwise

#### 4.41.3.13 expungeNote()

```
bool quantier::LocalStorageManager::expungeNote (
    Note & note,
    ErrorString & errorDescription )
```

expungeNote permanently deletes note from local storage.

Evernote API doesn't allow to delete notes from the remote storage, it can only be done by official desktop client or web client. So this method should be called only during the synchronization with remote database, when some note is found to be deleted via either official desktop client or web client.

#### Parameters

<i>note</i>	<a href="#">Note</a> to be expunged; may be changed as a result of the call, filled with fields like local uid or notebook guid or local uid
<i>errorDescription</i>	Error description if note could not be expunged

#### Returns

True if note was expunged successfully, false otherwise

#### 4.41.3.14 expungeNotebook()

```
bool quantier::LocalStorageManager::expungeNotebook (
    Notebook & notebook,
    ErrorString & errorDescription )
```

expungeNotebook permanently deletes the specified notebook from the local storage database.

Evernote API doesn't allow to delete the notebooks from the remote storage, it can only be done by the official desktop Evernote client or via its web client. So this method should be called only during the synchronization with the remote storage, when some notebook is found to be deleted via either the official desktop client or via the web client; also, this method can be called for local notebooks not synchronized with Evernote at all.

#### Parameters

<i>notebook</i>	The notebook to be expunged. Must have either "remote" guid or local uid set; the object is passed by reference and may be changed as a result of the call (filled with local uid if it was empty before the call)
<i>errorDescription</i>	Error description if the notebook could not be expunged

**Returns**

True if the notebook was expunged successfully, false otherwise

**4.41.3.15 expungeNotelessTagsFromLinkedNotebooks()**

```
bool quantier::LocalStorageManager::expungeNotelessTagsFromLinkedNotebooks (
    ErrorString & errorDescription )
```

expungeNotelessTagsFromLinkedNotebooks permanently deletes from the local storage database those tags which belong to some linked notebook and are not linked with any notes.

**Parameters**

<i>errorDescription</i>	Error description if tag could not be expunged
-------------------------	--

**Returns**

True if relevant tags were expunged successfully, false otherwise

**4.41.3.16 expungeSavedSearch()**

```
bool quantier::LocalStorageManager::expungeSavedSearch (
    SavedSearch & search,
    ErrorString & errorDescription )
```

expungeSavedSearch permanently deletes saved search from the local storage database.

**Parameters**

<i>search</i>	Saved search to be expunged; may be changed as a result of the call filled local uid if it was empty before the call
<i>errorDescription</i>	Error description if saved search could not be expunged

**Returns**

True if saved search was expunged successfully, false otherwise

**4.41.3.17 expungeTag()**

```
bool quantier::LocalStorageManager::expungeTag (
    Tag & tag,
```

```
QStringList & expungedChildTagLocalUids,
ErrorString & errorDescription )
```

expungeTag permanently deletes tag from the local storage database.

Evernote API doesn't allow to delete tags from remote storage, it can only be done by official desktop client or web client. So this method should be called only during the synchronization with remote database, when some tag is found to be deleted via either official desktop client or web client.

#### Parameters

<i>tag</i>	Tag to be expunged; may be changed as a result of the call, automatically filled with local uid if it was empty before the call
<i>expungedChildTagLocalUids</i>	If the expunged tag was a parent of some other tags, these were expunged as well; this parameter would contain the local uids of expunged child tags
<i>errorDescription</i>	Error description if tag could not be expunged

#### Returns

True if tag was expunged successfully, false otherwise

### 4.41.3.18 expungeUser()

```
bool quantier::LocalStorageManager::expungeUser (
    const User & user,
    ErrorString & errorDescription )
```

expungeUser permanently deletes the user from the local storage database

#### Parameters

<i>user</i>	The user to be expunged
<i>errorDescription</i>	Error description if the user could not be expunged

#### Returns

True if the user was expunged successfully, false otherwise

### 4.41.3.19 findDefaultNotebook()

```
bool quantier::LocalStorageManager::findDefaultNotebook (
    Notebook & notebook,
    ErrorString & errorDescription ) const
```

findDefaultNotebook attempts to find the default notebook in the local storage database.

**Parameters**

<i>notebook</i>	The default notebook to be found
<i>errorDescription</i>	Error description if the default notebook could not be found

**Returns**

True if the default notebook was found, false otherwise

**4.41.3.20 findDefaultOrLastUsedNotebook()**

```
bool quantier::LocalStorageManager::findDefaultOrLastUsedNotebook (
    Notebook & notebook,
    ErrorString & errorDescription ) const
```

findDefaultOrLastUsedNotebook attempts to find either the default or the last used notebook in the local storage database.

**Parameters**

<i>notebook</i>	Either the default or the last used notebook to be found
<i>errorDescription</i>	Error description if the default or the last used notebook could not be found

**Returns**

True if the default or the last used notebook were found, false otherwise

**4.41.3.21 findEnResource()**

```
bool quantier::LocalStorageManager::findEnResource (
    Resource & resource,
    const GetResourceOptions options,
    ErrorString & errorDescription ) const
```

findEnResource method attempts to find resource in the local storage database

**Parameters**

<i>resource</i>	<a href="#">Resource</a> to be found in the local storage database. If it has the "remote" Evernote service's guid set, this guid is used to identify the resource in the local storage database. Otherwise resource's local uid is used
<i>options</i>	Options specifying which optionally includable fields of the resource should actually be included
<i>errorDescription</i>	Error description if resource could not be found

**Returns**

True if resource was found successfully, false otherwise

**4.41.3.22 findLastUsedNotebook()**

```
bool quantier::LocalStorageManager::findLastUsedNotebook (
    Notebook & notebook,
    ErrorString & errorDescription ) const
```

findLastUsedNotebook attempts to find the last used notebook in the local storage database.

**Parameters**

<i>notebook</i>	The last used notebook to be found
<i>errorDescription</i>	Error description if the last used notebook could not be found

**Returns**

True if the last used notebook was found, false otherwise

**4.41.3.23 findLinkedNotebook()**

```
bool quantier::LocalStorageManager::findLinkedNotebook (
    LinkedNotebook & linkedNotebook,
    ErrorString & errorDescription ) const
```

findLinkedNotebook attempts to find and set all found fields for passed in by reference [LinkedNotebook](#) object. For [LinkedNotebook](#) local uid doesn't mean anything because it can only be considered valid if it has "remote" Evernote service's guid set. So this passed in [LinkedNotebook](#) object must have guid set to identify the linked notebook in the local storage database.

**Parameters**

<i>linkedNotebook</i>	Linked notebook to be found. Must have "remote" guid set
<i>errorDescription</i>	Error description if linked notebook could not be found

**Returns**

True if linked notebook was found, false otherwise

#### 4.41.3.24 findNote()

```
bool quentier::LocalStorageManager::findNote (
    Note & note,
    const GetNoteOptions options,
    ErrorString & errorDescription ) const
```

findNote - attempts to find note in the local storage database

##### Parameters

<i>note</i>	- note to be found in the local storage database. Must have either local or "remote" Evernote service's guid set
<i>options</i>	- options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	- error description if note could not be found

##### Returns

true if note was found successfully, false otherwise

#### 4.41.3.25 findNotebook()

```
bool quentier::LocalStorageManager::findNotebook (
    Notebook & notebook,
    ErrorString & errorDescription ) const
```

findNotebook attempts to find and set all found fields of the passed in [Notebook](#) object

If "remote" Evernote service's guid for the notebook is set, it is used to identify the notebook in the local storage database. Otherwise the notebook is identified by its local uid. If it's empty, the search would attempt to find the notebook by its name. If the name is also not set, the search would attempt to find the notebook by linked notebook guid assuming that no more than one notebook corresponds to the linked notebook guid. If linked notebook guid is also not set, the search would fail.

Important! Due to the fact that the notebook name is only unique within the users's own account as well as within each linked notebook, the result of the search by name depends on the notebook's linked notebook guid: if it is not set, the search by name would only search for the notebook with the specified name within the user's own account. If it is set, the search would only consider the linked notebook with the corresponding guid.

##### Parameters

<i>notebook</i>	The notebook to be found. Must have either "remote" or local uid or name or linked notebook guid set
<i>errorDescription</i>	Error description if the notebook could not be found

##### Returns

True if the notebook was found, false otherwise

#### 4.41.3.26 findNoteLocalUidsWithSearchQuery()

```
QStringList quantier::LocalStorageManager::findNoteLocalUidsWithSearchQuery (
    const NoteSearchQuery & noteSearchQuery,
    ErrorString & errorDescription ) const
```

findNoteLocalUidsWithSearchQuery attempts to find note local uids of notes corresponding to the passed in [NoteSearchQuery](#) object.

##### Parameters

<i>noteSearchQuery</i>	Filled <a href="#">NoteSearchQuery</a> object used to filter the notes
<i>errorDescription</i>	Error description in case note local uids could not be listed

##### Returns

The list of found notes' local uids or empty list in case of error

#### 4.41.3.27 findNotesWithSearchQuery()

```
NoteList quantier::LocalStorageManager::findNotesWithSearchQuery (
    const NoteSearchQuery & noteSearchQuery,
    const GetNoteOptions options,
    ErrorString & errorDescription ) const
```

findNotesWithSearchQuery attempts to find notes corresponding to the passed in [NoteSearchQuery](#) object.

##### Parameters

<i>noteSearchQuery</i>	Filled <a href="#">NoteSearchQuery</a> object used to filter the notes
<i>options</i>	Options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	Error description in case notes could not be listed

##### Returns

Either list of notes per [NoteSearchQuery](#) or empty list in case of error or no notes presence for the given [NoteSearchQuery](#)

#### 4.41.3.28 findSavedSearch()

```
bool quantier::LocalStorageManager::findSavedSearch (
    SavedSearch & search,
    ErrorString & errorDescription ) const
```

findSavedSearch attempts to find and fill the fields of passed in saved search object.

If "remote" Evernote services's guid for the saved search is set, it would be used to identify the saved search in the local storage. Otherwise the local uid would be used. If neither guid not local uid are set, saved search's name would be used. If the name is also not set, the search for saved search would fail.

## Parameters

<i>search</i>	<a href="#">SavedSearch</a> to be found in the local storage database
<i>errorDescription</i>	Error description if <a href="#">SavedSearch</a> could not be found

## Returns

True if [SavedSearch](#) was found, false otherwise

**4.41.3.29 findTag()**

```
bool quantier::LocalStorageManager::findTag (
    Tag & tag,
    ErrorString & errorDescription ) const
```

findTag attempts to find and fill the fields of passed in tag object.

If "remote" Evernote service's guid for the tag is set, it would be used to identify the tag in the local storage database. Otherwise the local uid would be used. If neither guid nor local uid are set, tag's name would be used. If the name is also not set, the search would fail.

Important! Due to the fact that the tag name is only unique within the users's own account as well as within each linked notebook, the result of the search by name depends on the tag's linked notebook guid: if it is not set, the search by name would only search for the tag with the specified name within the user's own account. If it is set, the search would only consider tags from a linked notebook with the corresponding guid.

## Parameters

<i>tag</i>	<a href="#">Tag</a> to be found in the local storage database; must have either guid, local uid or name set
<i>errorDescription</i>	Error description in case tag could not be found

## Returns

True if tag was found, false otherwise

**4.41.3.30 findUser()**

```
bool quantier::LocalStorageManager::findUser (
    User & user,
    ErrorString & errorDescription ) const
```

findUser attempts to find and fill the fields of the passed in [User](#) object which must have "id" field set as this value is used as the identifier of [User](#) objects in the local storage database



## Parameters

<i>user</i>	The user to be found. Must have "id" field set
<i>errorDescription</i>	Error description if the user could not be found

## Returns

True if the user was found successfully, false otherwise

**4.41.3.31 highestSupportedLocalStorageVersion()**

```
qint32 quantier::LocalStorageManager::highestSupportedLocalStorageVersion ( ) const
```

highestSupportedLocalStorageVersion returns the highest version of local storage persistence which the current build of libquantier is capable of working with

## Returns

Highest supported local storage version

**4.41.3.32 isLocalStorageVersionTooHigh()**

```
bool quantier::LocalStorageManager::isLocalStorageVersionTooHigh (
    ErrorString & errorDescription )
```

isLocalStorageVersionTooHigh method checks whether the existing local storage persistence has version which is too high for the currently run version of libquantier to work with i.e. whether the local storage has already been upgraded using a new version of libquantier.

NOTE: it is libquantier client code's responsibility to call this method and/or localStorageRequiresUpgrade method, libquantier won't call any of these on its own and will just attempt to work with the existing local storage, whatever version it is of. If version is too high, things can fail in most mysterious way, so the client code is obliged to call these methods to ensure the local storage version is checked properly.

## Parameters

<i>errorDescription</i>	Textual description of the error if the method was unable to determine whether the local storage version is too high for the currently run version of libquantier to work with, otherwise this parameter is not touched by the method
-------------------------	---

## Returns

True if local storage version is too high for the currently run version of libquantier to work with, false otherwise

#### 4.41.3.33 linkedNotebookCount()

```
int quentier::LocalStorageManager::linkedNotebookCount (
    QString & errorDescription ) const
```

linkedNotebookCount returns the number of linked notebooks stored in the local storage database.

##### Parameters

<i>errorDescription</i>	Error description if the number of linked notebooks count not be returned
-------------------------	---

##### Returns

Either non-negative number of linked notebooks or -1 if some error has occurred

#### 4.41.3.34 listAllLinkedNotebooks()

```
QList<LinkedNotebook> quentier::LocalStorageManager::listAllLinkedNotebooks (
    QString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListLinkedNotebooksOrder order = ListLinkedNotebooksOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending ) const
```

listAllLinkedNotebooks - attempts to list all linked notebooks within the account.

##### Parameters

<i>errorDescription</i>	Error description if linked notebooks could not be listed, otherwise this parameter is untouched
<i>limit</i>	Limit for the max number of linked notebooks in the result, zero by default which means no limit is set
<i>offset</i>	Number of linked notebooks to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of linked notebooks in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder

##### Returns

Either list of all linked notebooks or empty list in case of error or no linked notebooks presence within the account

#### 4.41.3.35 listAllNotebooks()

```
QList<Notebook> quentier::LocalStorageManager::listAllNotebooks (
    QString & errorDescription,
```

```

const size_t limit = 0,
const size_t offset = 0,
const ListNotebooksOrder order = ListNotebooksOrder::NoOrder,
const OrderDirection orderDirection = OrderDirection::Ascending,
const QString & linkedNotebookGuid = QString() ) const

```

listAllNotebooks attempts to list all notebooks within the current account from the local storage database.

#### Parameters

<i>errorDescription</i>	Error description if all notebooks could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	The limit for the max number of notebooks in the result, zero by default which means no limit is set
<i>offset</i>	The number of notebooks to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify a particular ordering of notebooks in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder
<i>linkedNotebookGuid</i>	If it's null, the method would list the notebooks ignoring their belonging to the current account or to some linked notebook; if it's empty, only the non-linked notebooks would be listed; otherwise, the only one notebook from the corresponding linked notebook would be listed

#### Returns

Either the list of all notebooks within the account or empty list in cases of error or no notebooks presence within the account

#### 4.41.3.36 listAllSavedSearches()

```

QList<SavedSearch> quantier::LocalStorageManager::listAllSavedSearches (
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListSavedSearchesOrder order = ListSavedSearchesOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending ) const

```

listAllSavedSearches lists all saved searches within the account.

#### Parameters

<i>errorDescription</i>	Error description if all saved searches could not be listed; otherwise this parameter is untouched
<i>limit</i>	Limit for the max number of saved searches in the result, zero by default which means no limit is set
<i>offset</i>	Number of saved searches to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of saved searches in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder

**Returns**

Either the list of all saved searches within the account or empty list in case of error or if there are no saved searches within the account

**4.41.3.37 listAllSharedNotebooks()**

```
QList<SharedNotebook> quentier::LocalStorageManager::listAllSharedNotebooks (
    ErrorString & errorDescription ) const
```

listAllSharedNotebooks attempts to list all shared notebooks within the account.

**Parameters**

<i>errorDescription</i>	Error description if shared notebooks could not be listed; if no error happens, this parameter is untouched
-------------------------	---

**Returns**

Either the list of all shared notebooks within the account or empty list in cases of error or no shared notebooks presence within the account

**4.41.3.38 listAllTags()**

```
QList<Tag> quentier::LocalStorageManager::listAllTags (
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListTagsOrder order = ListTagsOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending,
    const QString & linkedNotebookGuid = QString() ) const
```

listAllTags lists all tags within the current user's account.

**Parameters**

<i>errorDescription</i>	Error description if tags were not listed successfully. In such case the returned list of tags would be empty and error description won't be empty. However, if, for example, the list of tags is empty and error description is empty too, it means the current account does not have any tags created.
<i>limit</i>	Limit for the max number of tags in the result, zero by default which means no limit is set
<i>offset</i>	Number of tags to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of tags in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder
<i>linkedNotebookGuid</i>	If it's null, the method would list tags ignoring their belonging to the current account or to some linked notebook; if it's empty, only the tags from user's own account would be listed; otherwise, only the tags corresponding to the certain linked notebook would be listed

## Returns

The list of found tags within the account

## 4.41.3.39 listAllTagsPerNote()

```
QList<Tag> quantier::LocalStorageManager::listAllTagsPerNote (
    const Note & note,
    ErrorString & errorDescription,
    const ListObjectsOptions & flag = ListObjectsOption::ListAll,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListTagsOrder & order = ListTagsOrder::NoOrder,
    const OrderDirection & orderDirection = OrderDirection::Ascending ) const
```

listAllTagsPerNote lists all tags per given note

## Parameters

<i>note</i>	Note for which the list of tags is requested. If it has "remote" Evernote service's guid set, it is used to identify the note in the local storage database. Otherwise its local uid is used for that.
<i>errorDescription</i>	Error description if tags were not listed successfully. In such case the returned list of tags would be empty and error description won't be empty. However, if, for example, the list of tags is empty and error description is empty too, it means the provided note does not have any tags assigned to it.
<i>flag</i>	Input parameter used to set the filter for the desired tags to be listed
<i>limit</i>	Limit for the max number of tags in the result, zero by default which means no limit is set
<i>offset</i>	Number of tags to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of tags in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used;

## Returns

The list of found tags per note

## 4.41.3.40 listLinkedNotebooks()

```
QList<LinkedNotebook> quantier::LocalStorageManager::listLinkedNotebooks (
    const ListObjectsOptions flag,
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListLinkedNotebooksOrder order = ListLinkedNotebooksOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending ) const
```

listLinkedNotebooks attempts to list linked notebooks within the account according to the specified input flag.

## Parameters

<i>flag</i>	Input parameter used to set the filter for the desired linked notebooks to be listed
<i>errorDescription</i>	Error description if linked notebooks within the account could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	Limit for the max number of linked notebooks in the result, zero by default which means no limit is set
<i>offset</i>	Number of linked notebooks to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of linked notebooks in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder

## Returns

Either list of linked notebooks within the account conforming to the filter or empty list in cases of error or no linked notebooks conforming to the filter exist within the account

## 4.41.3.41 listNotebooks()

```
QList<Notebook> quentier::LocalStorageManager::listNotebooks (
    const ListObjectsOptions flag,
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListNotebooksOrder order = ListNotebooksOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending,
    const QString & linkedNotebookGuid = QString() ) const
```

listNotebooks attempts to list notebooks within the account according to the specified input flag

## Parameters

<i>flag</i>	Input parameter used to set the filter for the desired notebooks to be listed
<i>errorDescription</i>	Error description if notebooks within the account could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	The limit for the max number of notebooks in the result, zero by default which means no limit is set
<i>offset</i>	The number of notebooks to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify a particular ordering of notebooks in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder
<i>linkedNotebookGuid</i>	If it's null, the method would list notebooks ignoring their belonging to the current account or to some linked notebook; if it's empty, only the non-linked notebooks would be listed; otherwise, the only one notebook from the corresponding linked notebook would be listed

**Returns**

Either the list of notebooks within the account conforming to the filter or empty list in cases of error or no notebooks conforming to the filter exist within the account

**4.41.3.42 listNotes()**

```
QList<Note> quantier::LocalStorageManager::listNotes (
    const ListObjectsOptions flag,
    const GetNoteOptions options,
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListNotesOrder order = ListNotesOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending,
    const QString & linkedNotebookGuid = QString() ) const
```

listNotes attempts to list notes within the account according to the specified input flag.

**Parameters**

<i>flag</i>	Input parameter used to set the filter for the desired notes to be listed
<i>options</i>	Options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	Error description if notes within the account could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	Limit for the max number of notes in the result, zero by default which means no limit is set
<i>offset</i>	Number of notes to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of notes in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder
<i>linkedNotebookGuid</i>	If it's null, notes from both user's own notebooks and linked notebooks would be listed; if it's empty, only the notes from non-linked notebooks would be listed; otherwise, only the notes from the specified linked notebook would be listed

**Returns**

Either list of notes within the account conforming to the filter or empty list in cases of error or no notes conforming to the filter exist within the account

**4.41.3.43 listNotesByLocalUids()**

```
QList<Note> quantier::LocalStorageManager::listNotesByLocalUids (
    const QStringList & noteLocalUids,
    const GetNoteOptions options,
    ErrorString & errorDescription,
```

```

const ListObjectsOptions & flag = ListObjectsOption::ListAll,
const size_t limit = 0,
const size_t offset = 0,
const ListNotesOrder & order = ListNotesOrder::NoOrder,
const OrderDirection & orderDirection = OrderDirection::Ascending ) const

```

listNotesByLocalUids attempts to list notes given their local uids

The method would only return notes which it managed to find within the local storage i.e. having an invalid local uid in the list won't result in an error, just in the corresponding note not returned within the result

Notes within the result can be additionally filtered with flag parameter

#### Parameters

<i>noteLocalUids</i>	Local uids of notes to be listed
<i>options</i>	Options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	Error description in case notes could not be listed
<i>flag</i>	Input parameter used to set the filter for the desired notes to be listed
<i>limit</i>	Limit for the max number of notes in the result, zero by default which means no limit is set
<i>offset</i>	Number of notes to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of notes in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used;

#### Returns

Either list of notes by local uids or empty list in case of error or no notes corresponding to given local uids presence

#### 4.41.3.44 listNotesPerNotebook()

```

QList<Note> quantier::LocalStorageManager::listNotesPerNotebook (
    const Notebook & notebook,
    const GetNoteOptions options,
    ErrorString & errorDescription,
    const ListObjectsOptions & flag = ListObjectsOption::ListAll,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListNotesOrder & order = ListNotesOrder::NoOrder,
    const OrderDirection & orderDirection = OrderDirection::Ascending ) const

```

listNotesPerNotebook attempts to list notes per given notebook

#### Parameters

<i>notebook</i>	<a href="#">Notebook</a> for which the list of notes is requested. If it has the "remote" Evernote service's guid set, it would be used to identify the notebook in the local storage database, otherwise its local uid would be used
<i>options</i>	Options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	Error description in case notes could not be listed
<i>flag</i>	Input parameter used to set the filter for the desired notes to be listed



## Parameters

<i>limit</i>	Limit for the max number of notes in the result, zero by default which means no limit is set
<i>offset</i>	Number of notes to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of notes in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used;

## Returns

Either list of notes per notebook or empty list in case of error or no notes presence in the given notebook

## 4.41.3.45 listNotesPerNotebooksAndTags()

```
QList<Note> quantier::LocalStorageManager::listNotesPerNotebooksAndTags (
    const QStringList & notebookLocalUids,
    const QStringList & tagLocalUids,
    const GetNoteOptions options,
    ErrorString & errorDescription,
    const ListObjectsOptions & flag = ListObjectsOption::ListAll,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListNotesOrder & order = ListNotesOrder::NoOrder,
    const OrderDirection & orderDirection = OrderDirection::Ascending ) const
```

listNotesPerNotebooksAndTags attempts to list notes which are present within one of specified notebooks and are labeled with at least one of specified tags

## Parameters

<i>notebookLocalUids</i>	Local uids of notebooks to which the listed notes might belong
<i>tagLocalUids</i>	Local uids of tags with which the listed notes might be labeled
<i>options</i>	Options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	Error description in case notes could not be listed
<i>flag</i>	Input parameter used to set the filter for the desired notes to be listed
<i>limit</i>	Limit for the max number of notes in the result, zero by default which means no limit is set
<i>offset</i>	Number of notes to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of notes in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used;

## Returns

Either list of notes per notebooks and tags or empty list in case of error or no notes corresponding to given notebooks and tags presence

#### 4.41.3.46 listNotesPerTag()

```
QList<Note> quentier::LocalStorageManager::listNotesPerTag (
    const Tag & tag,
    const GetNoteOptions options,
    ErrorString & errorDescription,
    const ListObjectsOptions & flag = ListObjectsOption::ListAll,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListNotesOrder & order = ListNotesOrder::NoOrder,
    const OrderDirection & orderDirection = OrderDirection::Ascending ) const
```

listNotesPerTag attempts to list notes labeled with a given tag

##### Parameters

<i>tag</i>	Tag for which the list of notes labeled with it is requested. If it has the "remote" Evernote service's guid set, it is used to identify the tag in the local storage database, otherwise its local uid is used
<i>options</i>	Options specifying which optionally includable fields of the note should actually be included
<i>errorDescription</i>	Error description in case notes could not be listed
<i>flag</i>	Input parameter used to set the filter for the desired notes to be listed
<i>limit</i>	Limit for the max number of notes in the result, zero by default which means no limit is set
<i>offset</i>	Number of notes to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of notes in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used;

##### Returns

Either list of notes per tag or empty list in case of error or no notes labeled with the given tag presence

#### 4.41.3.47 listSavedSearches()

```
QList<SavedSearch> quentier::LocalStorageManager::listSavedSearches (
    const ListObjectsOptions flag,
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListSavedSearchesOrder order = ListSavedSearchesOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending ) const
```

listSavedSearches attempts to list saved searches within the account according to the specified input flag.

##### Parameters

<i>flag</i>	Input parameter used to set the filter for the desired saved searches to be listed
<i>errorDescription</i>	Error description if saved searches within the account could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	Limit for the max number of saved searches in the result, zero by default which means no limit is set
<i>offset</i>	Number of saved searches to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of saved searches in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder

## Returns

Either list of saved searches within the account conforming to the filter or empty list in cases of error or no saved searches conforming to the filter exist within the account

## 4.41.3.48 listSharedNotebooksPerNotebookGuid()

```
QList<SharedNotebook> quentier::LocalStorageManager::listSharedNotebooksPerNotebookGuid (
    const QString & notebookGuid,
    ErrorString & errorDescription ) const
```

listSharedNotebooksPerNotebookGuid - attempts to list all shared notebooks per given notebook's remote guid (not local uid, it's important).

## Parameters

<i>notebookGuid</i>	Remote Evernote service's guid of the notebook for which the shared notebooks are requested
<i>errorDescription</i>	Error description if shared notebooks per notebook guid could not be listed; if no error happens, this parameter is untouched

## Returns

Either the list of shared notebooks per notebook guid or empty list in case of error or no shared notebooks presence per given notebook guid

## 4.41.3.49 listTags()

```
QList<Tag> quentier::LocalStorageManager::listTags (
    const ListObjectsOptions flag,
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListTagsOrder & order = ListTagsOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending,
    const QString & linkedNotebookGuid = QString() ) const
```

listTags attempts to list tags within the account according to the specified input flag.

## Parameters

<i>flag</i>	Input parameter used to set the filter for the desired tags to be listed
<i>errorDescription</i>	Error description if notes within the account could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	Limit for the max number of tags in the result, zero by default which means no limit is set
<i>offset</i>	Number of tags to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of tags in the result, NoOrder by default

## Parameters

<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder
<i>linkedNotebookGuid</i>	If it's null, the method would list tags ignoring their belonging to the current account or to some linked notebook; if it's empty, only the tags from user's own account would be listed; otherwise, only the tags corresponding to the certain linked notebook would be listed

## Returns

Either list of tags within the account conforming to the filter or empty list in cases of error or no tags conforming to the filter exist within the account

## 4.41.3.50 listTagsWithNoteLocalUids()

```
QList<std::pair<Tag, QStringList> > quantier::LocalStorageManager::listTagsWithNoteLocalUids
(
    const ListObjectsOptions flag,
    ErrorString & errorDescription,
    const size_t limit = 0,
    const size_t offset = 0,
    const ListTagsOrder & order = ListTagsOrder::NoOrder,
    const OrderDirection orderDirection = OrderDirection::Ascending,
    const QString & linkedNotebookGuid = QString() ) const
```

listTagsWithNoteLocalUids attempts to list tags and their corresponding local uids within the account according to the specified input flag

The method is very similar to listTags only for each listed tag it returns the list of note local uids corresponding to notes labeled with the respective tag.

## Parameters

<i>flag</i>	Input parameter used to set the filter for the desired tags to be listed
<i>errorDescription</i>	Error description if notes within the account could not be listed; if no error happens, this parameter is untouched
<i>limit</i>	Limit for the max number of tags in the result, zero by default which means no limit is set
<i>offset</i>	Number of tags to skip in the beginning of the result, zero by default
<i>order</i>	Allows to specify particular ordering of tags in the result, NoOrder by default
<i>orderDirection</i>	Specifies the direction of ordering, by default ascending direction is used; this parameter has no meaning if order is equal to NoOrder
<i>linkedNotebookGuid</i>	If it's null, the method would list tags ignoring their belonging to the current account or to some linked notebook; if it's empty, only the tags from user's own account would be listed; otherwise, only the tags corresponding to the certain linked notebook would be listed

### Returns

Either list of tags and note local uids within the account conforming to the filter or empty list in cases of error or no tags conforming to the filter exist within the account

#### 4.41.3.51 localStorageRequiresUpgrade()

```
bool quantier::LocalStorageManager::localStorageRequiresUpgrade (
    ErrorString & errorDescription )
```

localStorageRequiresUpgrade method checks whether the existing local storage persistence requires to be upgraded. The upgrades may be required sometimes when new version of libquantier is rolled out which changes something in the internals of local storage organization. This method only checks for changes which are backwards incompatible i.e. once the local storage is upgraded, previous version of libquantier won't be able to work with it properly!

NOTE: it is libquantier client code's responsibility to call this method and/or isLocalStorageVersionTooHigh method, libquantier won't call any of these on its own and will just attempt to work with the existing local storage, whatever version it is of. If version is too high, things can fail in most mysterious way, so the client code is obliged to call these methods to ensure the local storage version is checked properly.

### Parameters

<i>errorDescription</i>	Textual description of the error if the method was unable to determine whether the local storage requires upgrade, otherwise this parameter is not touched by the method
-------------------------	--

### Returns

True if local storage requires upgrade, false otherwise

#### 4.41.3.52 localStorageVersion()

```
qint32 quantier::LocalStorageManager::localStorageVersion (
    ErrorString & errorDescription )
```

localStorageVersion method fetches the current version of local storage persistence which can be used for informational purposes.

### Parameters

<i>errorDescription</i>	Textual description of the error if the method was unable to determine the current version of local storage persistence
-------------------------	---

### Returns

Positive number indication local storage version or negative number in case of error retrieving the local storage version

#### 4.41.3.53 notebookCount()

```
int quentier::LocalStorageManager::notebookCount (
    ErrorString & errorDescription ) const
```

notebookCount returns the number of notebooks currently stored in the local storage database

##### Parameters

<i>errorDescription</i>	Error description if the number of notebooks could not be returned
-------------------------	--

##### Returns

Either non-negative value with the number of notebooks or -1 which means some error has occurred

#### 4.41.3.54 noteCount()

```
int quentier::LocalStorageManager::noteCount (
    ErrorString & errorDescription,
    const NoteCountOptions options = NoteCountOption::IncludeNonDeletedNotes ) const
```

noteCount returns the number of notes currently stored in the local storage database.

##### Parameters

<i>errorDescription</i>	Error description if the number of notes could not be returned
<i>options</i>	Options clarifying which notes to list; by default only non-deleted notes are listed

##### Returns

Either non-negative value with the number of notes or -1 which means some error occurred

#### 4.41.3.55 noteCountPerNotebook()

```
int quentier::LocalStorageManager::noteCountPerNotebook (
    const Notebook & notebook,
    ErrorString & errorDescription,
    const NoteCountOptions options = NoteCountOption::IncludeNonDeletedNotes ) const
```

noteCountPerNotebook returns the number of notes currently stored in the local storage database per given notebook.

## Parameters

<i>notebook</i>	<a href="#">Notebook</a> for which the number of notes is requested. If its guid is set, it is used to identify the notebook, otherwise its local uid is used
<i>errorDescription</i>	Error description if the number of notes per given notebook could not be returned
<i>options</i>	Options clarifying which notes to list; by default only non-deleted notes are listed

## Returns

Either non-negative value with the number of notes per given notebook or -1 which means some error occurred

**4.41.3.56 `noteCountPerNotebooksAndTags()`**

```
int quentier::LocalStorageManager::noteCountPerNotebooksAndTags (
    const QStringList & notebookLocalUids,
    const QStringList & tagLocalUids,
    ErrorString & errorDescription,
    const NoteCountOptions options = NoteCountOption::IncludeNonDeletedNotes ) const
```

`noteCountPerNotebooksAndTags` returns the number of notes currently stored in local storage database belonging to one of notebooks corresponding to given notebook local uids and labeled by at least one of tags corresponding to given tag local uids

## Parameters

<i>notebookLocalUids</i>	The list of notebook local uids used for filtering
<i>tagLocalUids</i>	The list of tag local uids used for filtering
<i>errorDescription</i>	Error description if the number of notes per notebooks and tags could not be returned
<i>options</i>	Options clarifying which notes to list; by default only non-deleted notes are listed

## Returns

Either non-negative value with the number of notes per given tag or -1 which means some error occurred

**4.41.3.57 `noteCountPerTag()`**

```
int quentier::LocalStorageManager::noteCountPerTag (
    const Tag & tag,
    ErrorString & errorDescription,
    const NoteCountOptions options = NoteCountOption::IncludeNonDeletedNotes ) const
```

`noteCountPerTag` returns the number of notes currently stored in local storage database labeled with given tag.

## Parameters

<i>tag</i>	Tag for which the number of notes labeled with it is requested. If its guid is set, it is used to identify the tag, otherwise its local uid is used
<i>errorDescription</i>	Error description if the number of notes per given tag could not be returned
<i>options</i>	Options clarifying which notes to list; by default only non-deleted notes are listed

## Returns

Either non-negative value with the number of notes per given tag or -1 which means some error occurred

**4.41.3.58 noteCountsPerAllTags()**

```
bool quentier::LocalStorageManager::noteCountsPerAllTags (
    QHash< QString, int > & noteCountsPerTagLocalUid,
    QString & errorDescription,
    const NoteCountOptions options = NoteCountOption::IncludeNonDeletedNotes ) const
```

noteCountsPerAllTags returns the number of notes currently stored in local storage database labeled with each tag stored in the local storage database.

## Parameters

<i>noteCountsPerTagLocalUid</i>	The result hash: note counts by tag local uids
<i>errorDescription</i>	Error description if the number of notes per all tags could not be returned
<i>options</i>	Options clarifying which notes to list; by default only non-deleted notes are listed

## Returns

True if note counts for all tags were computed successfully, false otherwise

**4.41.3.59 requiredLocalStoragePatches()**

```
QVector<std::shared_ptr<ILocalStoragePatch> > quentier::LocalStorageManager::requiredLocal↵
StoragePatches ( )
```

requiredLocalStoragePatches provides the client code with the list of patches which need to be applied to the current state of local storage in order to bring it to a state compatible with the current version of code. If no patches are required, an empty list of patches is returned.

The client code should apply each patch in the exact order in which they are returned by this method.

## Returns

The vector of patches required to be applied to the current local storage version



#### 4.41.3.60 savedSearchCount()

```
int quantier::LocalStorageManager::savedSearchCount (
    ErrorString & errorDescription ) const
```

savedSearchCount returns the number of saved seacrhes currently stored in local storage database.

##### Parameters

<i>errorDescription</i>	Error description if the number of saved seacrhes could not be returned
-------------------------	---

##### Returns

Either non-negative value with the number of saved seacrhes or -1 which means some error occurred

#### 4.41.3.61 switchUser()

```
void quantier::LocalStorageManager::switchUser (
    const Account & account,
    const StartupOptions options = {} )
```

switchUser - switches to another local storage database file associated with the passed in account

If optional "startFromScratch" parameter is set to true (it is false by default), the database file would be erased and only then - opened. If optional "overrideLock" parameter is set to true, the advisory lock set on the database file (if any) would be forcefully removed; otherwise, if this parameter if set to false, the presence of advisory lock on the database file woud cause the method to throw [DatabaseLockedException](#)

##### Parameters

<i>account</i>	The account to which the local storage is to be switched
<i>options</i>	Startup options for the local storage, none enabled by default

#### 4.41.3.62 tagCount()

```
int quantier::LocalStorageManager::tagCount (
    ErrorString & errorDescription ) const
```

tagCount returns the number of non-deleted tags currently stored in the local storage database.

##### Parameters

<i>errorDescription</i>	Error description if the number of tags could not be returned
-------------------------	---

**Returns**

Either non-negative value with the number of tags or -1 which means some error occurred

**4.41.3.63 updateEnResource()**

```
bool quentier::LocalStorageManager::updateEnResource (
    Resource & resource,
    ErrorString & errorDescription )
```

updateEnResource updates passed in resource in the local storage database.

If the resource has "remote" Evernote service's guid set, it is identified by this guid in the local storage database. If no resource with such guid is found, the local uid is used to identify the resource in the local storage database. If the resource has no guid, the local uid is used to identify it in the local storage database.

**Parameters**

<i>resource</i>	<a href="#">Resource</a> to be updated; may be changed as a result of the call, automatically filled with local uid and note local uid and/or guid if these were empty before the call
<i>errorDescription</i>	Error description if resource could not be updated

**Returns**

True if resource was updated successfully, false otherwise

**4.41.3.64 updateLinkedNotebook()**

```
bool quentier::LocalStorageManager::updateLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    ErrorString & errorDescription )
```

updateLinkedNotebook updates passed in [LinkedNotebook](#) in the local storage database; [LinkedNotebook](#) must have "remote" Evernote service's guid set.

**Parameters**

<i>linkedNotebook</i>	<a href="#">LinkedNotebook</a> to be updated in the local storage database
<i>errorDescription</i>	Error description if linked notebook could not be updated

**Returns**

True if linked notebook was updated successfully, false otherwise

**4.41.3.65 updateNote()**

```
bool quentier::LocalStorageManager::updateNote (
    Note & note,
    const UpdateNoteOptions options,
    ErrorString & errorDescription )
```

updateNote updates passed in [Note](#) in the local storage database.

If the note has "remote" Evernote service's guid set, it is identified by this guid in the local storage database. If no note with such guid is found, the local uid is used to identify the note in the local storage database. If the note has no guid, the local uid is used to identify it in the local storage database.

A special way in which this method might be used is the update of a note which clears note's guid. This way is special because it imposes certain requirements onto the resources which the note might have. However, it is only relevant if options input parameter has UpdateResourceMetadata flag enabled. The requirements for this special case are as follows:

- each resource should not have noteGuid field set to a non-empty value
- each resource should not have guid field set to a non-empty value as it makes no sense for note without guid i.e. note not synchronized with Evernote to own a resource which has guid i.e. is synchronized with Evernote

**Parameters**

<i>note</i>	<a href="#">Note</a> to be updated in the local storage database; required to contain either "remote" notebook guid or local notebook uid; may be changed as a result of the call, filled with fields like local uid or notebook guid or local uid if any of these were empty before the call; also tag guids are filled if the note passed in contained only tag local uids and tag local uids are filled if the note passed in contained only tag guids. Bear in mind that after the call the note may not have the representative resources if "updateNoteOptions" input parameter contained no "UpdateResourceMetadata" flag as well as it may not have the representative tags if "UpdateTags" flag was not set
<i>options</i>	Options specifying which optionally updatable fields of the note should actually be updated
<i>errorDescription</i>	Error description if note could not be updated

**Returns**

True if note was updated successfully, false otherwise

**4.41.3.66 updateNotebook()**

```
bool quentier::LocalStorageManager::updateNotebook (
    Notebook & notebook,
    ErrorString & errorDescription )
```

updateNotebook updates the passed in [Notebook](#) in the local storage database

If the notebook has "remote" Evernote service's guid set, it is identified by this guid in the local storage database. Otherwise it is identified by the local uid.

**Parameters**

<i>notebook</i>	<a href="#">Notebook</a> to be updated in the local storage database; the object is passed by reference and may be changed as a result of the call (filled with autocompleted fields like local uid if it was empty before the call)
<i>errorDescription</i>	Error description if the notebook could not be updated

**Returns**

True if the notebook was updated successfully, false otherwise

**4.41.3.67 updateSavedSearch()**

```
bool quantier::LocalStorageManager::updateSavedSearch (
    SavedSearch & search,
    ErrorString & errorDescription )
```

updateSavedSearch updates passed in [SavedSearch](#) in the local storage database.

If search has "remote" Evernote service's guid set, it is identified in the database by this guid. If the saved search has no guid, the local uid is used to identify it in the local storage database.

**Parameters**

<i>search</i>	<a href="#">SavedSearch</a> filled with values to be updated in the local storage database; may be changed as a result of the call filled local uid if it was empty before the call
<i>errorDescription</i>	Error description if <a href="#">SavedSearch</a> could not be updated

**Returns**

True if [SavedSearch](#) was updated successfully, false otherwise

**4.41.3.68 updateTag()**

```
bool quantier::LocalStorageManager::updateTag (
    Tag & tag,
    ErrorString & errorDescription )
```

updateTag updates passed in [Tag](#) in the local storage database.

If the tag has "remote" Evernote service's guid set, it is identified by this guid in the local storage database. If the tag has no guid, the local uid is used to identify it in the local storage database.

## Parameters

<i>tag</i>	<a href="#">Tag</a> filled with values to be updated in the local storage database. <a href="#">Note</a> that it can be changed * as a result of the call: automatically filled with local uid if it was empty before the call
<i>errorDescription</i>	Error description if tag could not be updated

## Returns

True if tag was updated successfully, false otherwise

**4.41.3.69 updateUser()**

```
bool quantier::LocalStorageManager::updateUser (
    const User & user,
    ErrorString & errorDescription )
```

updateUser updates the passed in [User](#) object in the local storage database

The table with Users is only involved in operations with notebooks which have "contact" field set which in turn is used with business accounts

## Parameters

<i>user</i>	The user to be updated in the local storage database
<i>errorDescription</i>	Error description if the user could not be updated

## Returns

True if the user was updated successfully, false otherwise

**4.41.3.70 upgradeProgress**

```
void quantier::LocalStorageManager::upgradeProgress (
    double progress ) [signal]
```

[LocalStorageManager](#) is capable of performing automatic database upgrades if/when it is necessary.

As the database upgrade can be a lengthy operation, this signal is meant to provide some feedback on the progress of the upgrade

## Parameters

<i>progress</i>	The value from 0 to 1 denoting the database upgrade progress
-----------------	--

#### 4.41.3.71 userCount()

```
int quantier::LocalStorageManager::userCount (
    QString & errorDescription ) const
```

userCount returns the number of non-deleted users currently stored in the local storage database

##### Parameters

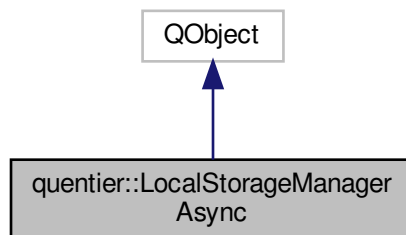
<i>errorDescription</i>	Error description if the number of users could not be returned
-------------------------	--

##### Returns

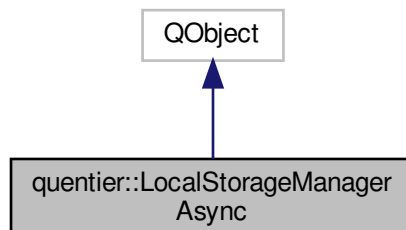
Either non-negative value with the number of users or -1 which means some error has occurred

## 4.42 quantier::LocalStorageManagerAsync Class Reference

Inheritance diagram for quantier::LocalStorageManagerAsync:



Collaboration diagram for quantier::LocalStorageManagerAsync:



## Public Slots

- void **init** ()
- void **onGetUserCountRequest** (QUuid requestId)
- void **onSwitchUserRequest** ([Account](#) account, LocalStorageManager::StartupOptions startupOptions, QUuid requestId)
- void **onAddUserRequest** ([User](#) user, QUuid requestId)
- void **onUpdateUserRequest** ([User](#) user, QUuid requestId)
- void **onFindUserRequest** ([User](#) user, QUuid requestId)
- void **onDeleteUserRequest** ([User](#) user, QUuid requestId)
- void **onExpungeUserRequest** ([User](#) user, QUuid requestId)
- void **onGetNotebookCountRequest** (QUuid requestId)
- void **onAddNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onUpdateNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onFindNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onFindDefaultNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onFindLastUsedNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onFindDefaultOrLastUsedNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onListAllNotebooksRequest** (size\_t limit, size\_t offset, [LocalStorageManager::ListNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QUuid requestId)
- void **onListAllSharedNotebooksRequest** (QUuid requestId)
- void **onListNotebooksRequest** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QUuid requestId)
- void **onListSharedNotebooksPerNotebookGuidRequest** (QString notebookGuid, QUuid requestId)
- void **onExpungeNotebookRequest** ([Notebook](#) notebook, QUuid requestId)
- void **onGetLinkedNotebookCountRequest** (QUuid requestId)
- void **onAddLinkedNotebookRequest** ([LinkedNotebook](#) linkedNotebook, QUuid requestId)
- void **onUpdateLinkedNotebookRequest** ([LinkedNotebook](#) linkedNotebook, QUuid requestId)
- void **onFindLinkedNotebookRequest** ([LinkedNotebook](#) linkedNotebook, QUuid requestId)
- void **onListAllLinkedNotebooksRequest** (size\_t limit, size\_t offset, [LocalStorageManager::ListLinkedNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onListLinkedNotebooksRequest** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListLinkedNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onExpungeLinkedNotebookRequest** ([LinkedNotebook](#) linkedNotebook, QUuid requestId)
- void **onGetNoteCountRequest** (LocalStorageManager::NoteCountOptions options, QUuid requestId)
- void **onGetNoteCountPerNotebookRequest** ([Notebook](#) notebook, LocalStorageManager::NoteCountOptions options, QUuid requestId)
- void **onGetNoteCountPerTagRequest** ([Tag](#) tag, LocalStorageManager::NoteCountOptions options, QUuid requestId)
- void **onGetNoteCountsPerAllTagsRequest** (LocalStorageManager::NoteCountOptions options, QUuid requestId)
- void **onGetNoteCountPerNotebooksAndTagsRequest** (QStringList notebookLocalUids, QStringList tagLocalUids, LocalStorageManager::NoteCountOptions options, QUuid requestId)
- void **onAddNoteRequest** ([Note](#) note, QUuid requestId)
- void **onUpdateNoteRequest** ([Note](#) note, LocalStorageManager::UpdateNoteOptions options, QUuid requestId)
- void **onFindNoteRequest** ([Note](#) note, LocalStorageManager::GetNoteOptions options, QUuid requestId)
- void **onListNotesPerNotebookRequest** ([Notebook](#) notebook, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onListNotesPerTagRequest** ([Tag](#) tag, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)

- void **onListNotesPerNotebooksAndTagsRequest** (QStringList notebookLocalUids, QStringList tagLocalUids, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onListNotesByLocalUidsRequest** (QStringList noteLocalUids, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onListNotesRequest** (LocalStorageManager::ListObjectsOptions flag, LocalStorageManager::GetNoteOptions options, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QUuid requestId)
- void **onFindNoteLocalUidsWithSearchQuery** ([NoteSearchQuery](#) noteSearchQuery, QUuid requestId)
- void **onExpungeNoteRequest** ([Note](#) note, QUuid requestId)
- void **onGetTagCountRequest** (QUuid requestId)
- void **onAddTagRequest** ([Tag](#) tag, QUuid requestId)
- void **onUpdateTagRequest** ([Tag](#) tag, QUuid requestId)
- void **onFindTagRequest** ([Tag](#) tag, QUuid requestId)
- void **onListAllTagsPerNoteRequest** ([Note](#) note, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onListAllTagsRequest** (size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QUuid requestId)
- void **onListTagsRequest** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QUuid requestId)
- void **onListTagsWithNoteLocalUidsRequest** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QUuid requestId)
- void **onExpungeTagRequest** ([Tag](#) tag, QUuid requestId)
- void **onExpungeNotelessTagsFromLinkedNotebooksRequest** (QUuid requestId)
- void **onGetResourceCountRequest** (QUuid requestId)
- void **onAddResourceRequest** ([Resource](#) resource, QUuid requestId)
- void **onUpdateResourceRequest** ([Resource](#) resource, QUuid requestId)
- void **onFindResourceRequest** ([Resource](#) resource, LocalStorageManager::GetResourceOptions options, QUuid requestId)
- void **onExpungeResourceRequest** ([Resource](#) resource, QUuid requestId)
- void **onGetSavedSearchCountRequest** (QUuid requestId)
- void **onAddSavedSearchRequest** ([SavedSearch](#) search, QUuid requestId)
- void **onUpdateSavedSearchRequest** ([SavedSearch](#) search, QUuid requestId)
- void **onFindSavedSearchRequest** ([SavedSearch](#) search, QUuid requestId)
- void **onListAllSavedSearchesRequest** (size\_t limit, size\_t offset, [LocalStorageManager::ListSavedSearchesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onListSavedSearchesRequest** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListSavedSearchesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **onExpungeSavedSearchRequest** ([SavedSearch](#) search, QUuid requestId)
- void **onAccountHighUsnRequest** (QString linkedNotebookGuid, QUuid requestId)

## Signals

- void **initialized** ()
- void **getUserCountComplete** (int userCount, QUuid requestId)
- void **getUserCountFailed** ([ErrorString](#) errorDescription, QUuid requestId)
- void **switchUserComplete** ([Account](#) account, QUuid requestId)
- void **switchUserFailed** ([Account](#) account, [ErrorString](#) errorDescription, QUuid requestId)
- void **addUserComplete** ([User](#) user, QUuid requestId)



- void **addUserFailed** ([User](#) user, [ErrorString](#) errorDescription, QUuid requestId)
- void **updateUserComplete** ([User](#) user, QUuid requestId)
- void **updateUserFailed** ([User](#) user, [ErrorString](#) errorDescription, QUuid requestId)
- void **findUserComplete** ([User](#) foundUser, QUuid requestId)
- void **findUserFailed** ([User](#) user, [ErrorString](#) errorDescription, QUuid requestId)
- void **deleteUserComplete** ([User](#) user, QUuid requestId)
- void **deleteUserFailed** ([User](#) user, [ErrorString](#) errorDescription, QUuid requestId)
- void **expungeUserComplete** ([User](#) user, QUuid requestId)
- void **expungeUserFailed** ([User](#) user, [ErrorString](#) errorDescription, QUuid requestId)
- void **getNotebookCountComplete** (int notebookCount, QUuid requestId)
- void **getNotebookCountFailed** ([ErrorString](#) errorDescription, QUuid requestId)
- void **addNotebookComplete** ([Notebook](#) notebook, QUuid requestId)
- void **addNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **updateNotebookComplete** ([Notebook](#) notebook, QUuid requestId)
- void **updateNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **findNotebookComplete** ([Notebook](#) foundNotebook, QUuid requestId)
- void **findNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **findDefaultNotebookComplete** ([Notebook](#) foundNotebook, QUuid requestId)
- void **findDefaultNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **findLastUsedNotebookComplete** ([Notebook](#) foundNotebook, QUuid requestId)
- void **findLastUsedNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **findDefaultOrLastUsedNotebookComplete** ([Notebook](#) foundNotebook, QUuid requestId)
- void **findDefaultOrLastUsedNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **listAllNotebooksComplete** (size\_t limit, size\_t offset, [LocalStorageManager::ListNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QList< [Notebook](#) > foundNotebooks, QUuid requestId)
- void **listAllNotebooksFailed** (size\_t limit, size\_t offset, [LocalStorageManager::ListNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, [ErrorString](#) errorDescription, QUuid requestId)
- void **listNotebooksComplete** ([LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QList< [Notebook](#) > foundNotebooks, QUuid requestId)
- void **listNotebooksFailed** ([LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, [ErrorString](#) errorDescription, QUuid requestId)
- void **listAllSharedNotebooksComplete** (QList< [SharedNotebook](#) > foundSharedNotebooks, QUuid requestId)
- void **listAllSharedNotebooksFailed** ([ErrorString](#) errorDescription, QUuid requestId)
- void **listSharedNotebooksPerNotebookGuidComplete** (QString notebookGuid, QList< [SharedNotebook](#) > foundSharedNotebooks, QUuid requestId)
- void **listSharedNotebooksPerNotebookGuidFailed** (QString notebookGuid, [ErrorString](#) errorDescription, QUuid requestId)
- void **expungeNotebookComplete** ([Notebook](#) notebook, QUuid requestId)
- void **expungeNotebookFailed** ([Notebook](#) notebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **getLinkedNotebookCountComplete** (int linkedNotebookCount, QUuid requestId)
- void **getLinkedNotebookCountFailed** ([ErrorString](#) errorDescription, QUuid requestId)
- void **addLinkedNotebookComplete** ([LinkedNotebook](#) linkedNotebook, QUuid requestId)
- void **addLinkedNotebookFailed** ([LinkedNotebook](#) linkedNotebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **updateLinkedNotebookComplete** ([LinkedNotebook](#) linkedNotebook, QUuid requestId)
- void **updateLinkedNotebookFailed** ([LinkedNotebook](#) linkedNotebook, [ErrorString](#) errorDescription, QUuid requestId)
- void **findLinkedNotebookComplete** ([LinkedNotebook](#) foundLinkedNotebook, QUuid requestId)
- void **findLinkedNotebookFailed** ([LinkedNotebook](#) linkedNotebook, [ErrorString](#) errorDescription, QUuid requestId)

- void **listAllLinkedNotebooksComplete** (size\_t limit, size\_t offset, [LocalStorageManager::ListLinkedNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [QList< LinkedNotebook >](#) foundLinkedNotebooks, [QUuid](#) requestId)
- void **listAllLinkedNotebooksFailed** (size\_t limit, size\_t offset, [LocalStorageManager::ListLinkedNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **listLinkedNotebooksComplete** ([LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListLinkedNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [QList< LinkedNotebook >](#) foundLinkedNotebooks, [QUuid](#) requestId)
- void **listLinkedNotebooksFailed** ([LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListLinkedNotebooksOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **expungeLinkedNotebookComplete** ([LinkedNotebook](#) linkedNotebook, [QUuid](#) requestId)
- void **expungeLinkedNotebookFailed** ([LinkedNotebook](#) linkedNotebook, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **getNoteCountComplete** (int noteCount, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountFailed** ([ErrorString](#) errorDescription, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountPerNotebookComplete** (int noteCount, [Notebook](#) notebook, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountPerNotebookFailed** ([ErrorString](#) errorDescription, [Notebook](#) notebook, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountPerTagComplete** (int noteCount, [Tag](#) tag, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountPerTagFailed** ([ErrorString](#) errorDescription, [Tag](#) tag, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountsPerAllTagsComplete** ([QHash< QString, int >](#) noteCountsPerTagLocalUids, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountsPerAllTagsFailed** ([ErrorString](#) errorDescription, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountPerNotebooksAndTagsComplete** (int noteCount, [QStringList](#) notebookLocalUids, [QStringList](#) tagLocalUids, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **getNoteCountPerNotebooksAndTagsFailed** ([ErrorString](#) errorDescription, [QStringList](#) notebookLocalUids, [QStringList](#) tagLocalUids, [LocalStorageManager::NoteCountOptions](#) options, [QUuid](#) requestId)
- void **addNoteComplete** ([Note](#) note, [QUuid](#) requestId)
- void **addNoteFailed** ([Note](#) note, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **updateNoteComplete** ([Note](#) note, [LocalStorageManager::UpdateNoteOptions](#) options, [QUuid](#) requestId)
- void **updateNoteFailed** ([Note](#) note, [LocalStorageManager::UpdateNoteOptions](#) options, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **findNoteComplete** ([Note](#) foundNote, [LocalStorageManager::GetNoteOptions](#) options, [QUuid](#) requestId)
- void **findNoteFailed** ([Note](#) note, [LocalStorageManager::GetNoteOptions](#) options, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **listNotesPerNotebookComplete** ([Notebook](#) notebook, [LocalStorageManager::GetNoteOptions](#) options, [LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [QList< Note >](#) foundNotes, [QUuid](#) requestId)
- void **listNotesPerNotebookFailed** ([Notebook](#) notebook, [LocalStorageManager::GetNoteOptions](#) options, [LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, [QUuid](#) requestId)
- void **listNotesPerTagComplete** ([Tag](#) tag, [LocalStorageManager::GetNoteOptions](#) options, [LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [QList< Note >](#) foundNotes, [QUuid](#) requestId)
- void **listNotesPerTagFailed** ([Tag](#) tag, [LocalStorageManager::GetNoteOptions](#) options, [LocalStorageManager::ListObjectsOptions](#) flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, [QUuid](#) requestId)

- void **listNotesPerNotebooksAndTagsComplete** (QStringList notebookLocalUids, QStringList tagLocalUids, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QList< [Note](#) > foundNotes, QUuid requestId)
- void **listNotesPerNotebooksAndTagsFailed** (QStringList notebookLocalUids, QStringList tagLocalUids, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, QUuid requestId)
- void **listNotesByLocalUidsComplete** (QStringList noteLocalUids, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QList< [Note](#) > foundNotes, QUuid requestId)
- void **listNotesByLocalUidsFailed** (QStringList noteLocalUids, LocalStorageManager::GetNoteOptions options, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, QUuid requestId)
- void **listNotesComplete** (LocalStorageManager::ListObjectsOptions flag, LocalStorageManager::GetNoteOptions options, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QList< [Note](#) > foundNotes, QUuid requestId)
- void **listNotesFailed** (LocalStorageManager::ListObjectsOptions flag, LocalStorageManager::GetNoteOptions options, size\_t limit, size\_t offset, [LocalStorageManager::ListNotesOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, [ErrorString](#) errorDescription, QUuid requestId)
- void **findNoteLocalUidsWithSearchQueryComplete** (QStringList noteLocalUids, [NoteSearchQuery](#) noteSearchQuery, QUuid requestId)
- void **findNoteLocalUidsWithSearchQueryFailed** ([NoteSearchQuery](#) noteSearchQuery, [ErrorString](#) errorDescription, QUuid requestId)
- void **expungeNoteComplete** ([Note](#) note, QUuid requestId)
- void **expungeNoteFailed** ([Note](#) note, [ErrorString](#) errorDescription, QUuid requestId)
- void **noteMovedToAnotherNotebook** (QString noteLocalUid, QString previousNotebookLocalUid, QString newNotebookLocalUid)
- void **noteTagListChanged** (QString noteLocalUid, QStringList previousNoteTagLocalUids, QStringList newNoteTagLocalUids)
- void **getTagCountComplete** (int tagCount, QUuid requestId)
- void **getTagCountFailed** ([ErrorString](#) errorDescription, QUuid requestId)
- void **addTagComplete** ([Tag](#) tag, QUuid requestId)
- void **addTagFailed** ([Tag](#) tag, [ErrorString](#) errorDescription, QUuid requestId)
- void **updateTagComplete** ([Tag](#) tag, QUuid requestId)
- void **updateTagFailed** ([Tag](#) tag, [ErrorString](#) errorDescription, QUuid requestId)
- void **linkTagWithNoteComplete** ([Tag](#) tag, [Note](#) note, QUuid requestId)
- void **linkTagWithNoteFailed** ([Tag](#) tag, [Note](#) note, [ErrorString](#) errorDescription, QUuid requestId)
- void **findTagComplete** ([Tag](#) tag, QUuid requestId)
- void **findTagFailed** ([Tag](#) tag, [ErrorString](#) errorDescription, QUuid requestId)
- void **listAllTagsPerNoteComplete** (QList< [Tag](#) > foundTags, [Note](#) note, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QUuid requestId)
- void **listAllTagsPerNoteFailed** ([Note](#) note, LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, [ErrorString](#) errorDescription, QUuid requestId)
- void **listAllTagsComplete** (size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QList< [Tag](#) > foundTags, QUuid requestId)
- void **listAllTagsFailed** (size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, [ErrorString](#) errorDescription, QUuid requestId)
- void **listTagsComplete** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, [LocalStorageManager::ListTagsOrder](#) order, [LocalStorageManager::OrderDirection](#) orderDirection, QString linkedNotebookGuid, QList< [Tag](#) > foundTags, QUuid requestId=QUuid())

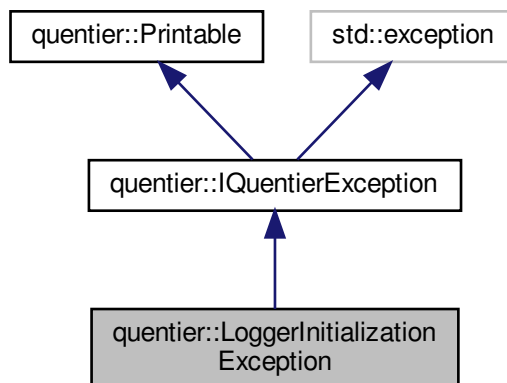
- void **listTagsFailed** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, LocalStorageManager::ListTagsOrder order, LocalStorageManager::OrderDirection orderDirection, QString linkedNotebookGuid, ErrorString errorDescription, QUuid requestId)
- void **listTagsWithNoteLocalUidsComplete** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, LocalStorageManager::ListTagsOrder order, LocalStorageManager::OrderDirection orderDirection, QString linkedNotebookGuid, QList< std::pair< Tag, QStringList >> foundTags, QUuid requestId)
- void **listTagsWithNoteLocalUidsFailed** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, LocalStorageManager::ListTagsOrder order, LocalStorageManager::OrderDirection orderDirection, QString linkedNotebookGuid, ErrorString errorDescription, QUuid requestId)
- void **expungeTagComplete** (Tag tag, QStringList expungedChildTagLocalUids, QUuid requestId)
- void **expungeTagFailed** (Tag tag, ErrorString errorDescription, QUuid requestId)
- void **expungeNotelessTagsFromLinkedNotebooksComplete** (QUuid requestId)
- void **expungeNotelessTagsFromLinkedNotebooksFailed** (ErrorString errorDescription, QUuid requestId)
- void **getResourceCountComplete** (int resourceCount, QUuid requestId)
- void **getResourceCountFailed** (ErrorString errorDescription, QUuid requestId)
- void **addResourceComplete** (Resource resource, QUuid requestId)
- void **addResourceFailed** (Resource resource, ErrorString errorDescription, QUuid requestId)
- void **updateResourceComplete** (Resource resource, QUuid requestId)
- void **updateResourceFailed** (Resource resource, ErrorString errorDescription, QUuid requestId)
- void **findResourceComplete** (Resource resource, LocalStorageManager::GetResourceOptions options, QUuid requestId)
- void **findResourceFailed** (Resource resource, LocalStorageManager::GetResourceOptions options, ErrorString errorDescription, QUuid requestId)
- void **expungeResourceComplete** (Resource resource, QUuid requestId)
- void **expungeResourceFailed** (Resource resource, ErrorString errorDescription, QUuid requestId)
- void **getSavedSearchCountComplete** (int savedSearchCount, QUuid requestId)
- void **getSavedSearchCountFailed** (ErrorString errorDescription, QUuid requestId)
- void **addSavedSearchComplete** (SavedSearch search, QUuid requestId)
- void **addSavedSearchFailed** (SavedSearch search, ErrorString errorDescription, QUuid requestId)
- void **updateSavedSearchComplete** (SavedSearch search, QUuid requestId)
- void **updateSavedSearchFailed** (SavedSearch search, ErrorString errorDescription, QUuid requestId)
- void **findSavedSearchComplete** (SavedSearch search, QUuid requestId)
- void **findSavedSearchFailed** (SavedSearch search, ErrorString errorDescription, QUuid requestId)
- void **listAllSavedSearchesComplete** (size\_t limit, size\_t offset, LocalStorageManager::ListSavedSearchesOrder order, LocalStorageManager::OrderDirection orderDirection, QList< SavedSearch > foundSearches, QUuid requestId)
- void **listAllSavedSearchesFailed** (size\_t limit, size\_t offset, LocalStorageManager::ListSavedSearchesOrder order, LocalStorageManager::OrderDirection orderDirection, ErrorString errorDescription, QUuid requestId)
- void **listSavedSearchesComplete** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, LocalStorageManager::ListSavedSearchesOrder order, LocalStorageManager::OrderDirection orderDirection, QList< SavedSearch > foundSearches, QUuid requestId)
- void **listSavedSearchesFailed** (LocalStorageManager::ListObjectsOptions flag, size\_t limit, size\_t offset, LocalStorageManager::ListSavedSearchesOrder order, LocalStorageManager::OrderDirection orderDirection, ErrorString errorDescription, QUuid requestId)
- void **expungeSavedSearchComplete** (SavedSearch search, QUuid requestId)
- void **expungeSavedSearchFailed** (SavedSearch search, ErrorString errorDescription, QUuid requestId)
- void **accountHighUsnComplete** (qint32 usn, QString linkedNotebookGuid, QUuid requestId)
- void **accountHighUsnFailed** (QString linkedNotebookGuid, ErrorString errorDescription, QUuid requestId)

## Public Member Functions

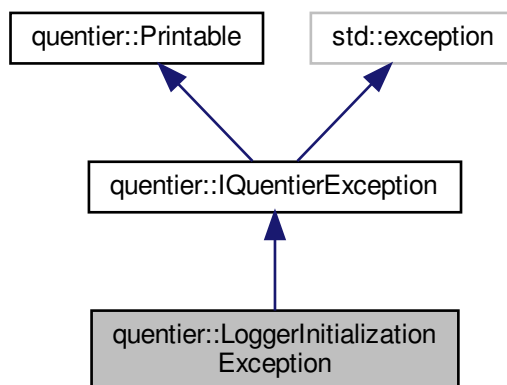
- **LocalStorageManagerAsync** (const Account &account, LocalStorageManager::StartupOptions options={}, QObject \*parent=nullptr)
- void **setUseCache** (const bool useCache)
- const LocalStorageCacheManager \* **localStorageCacheManager** () const
- bool **installCacheExpiryFunction** (const ILocalStorageCacheExpiryChecker &checker)
- const LocalStorageManager \* **localStorageManager** () const
- LocalStorageManager \* **localStorageManager** ()

## 4.43 quantier::LoggerInitializationException Class Reference

Inheritance diagram for quantier::LoggerInitializationException:



Collaboration diagram for quantier::LoggerInitializationException:



### Public Member Functions

- **LoggerInitializationException** (const [ErrorString](#) &message)

### Protected Member Functions

- const QString **exceptionDisplayName** () const override

## 4.44 quotient::LRUCache< Key, Value, Allocator > Class Template Reference

### Public Types

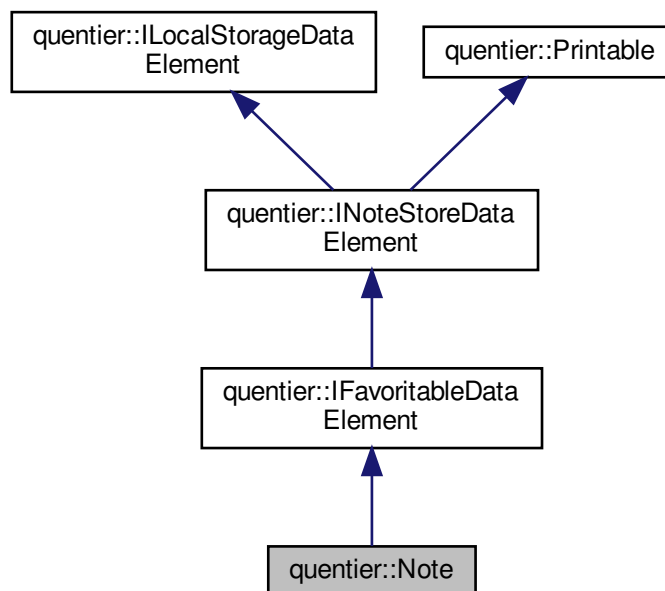
- using **key\_type** = Key
- using **mapped\_type** = Value
- using **allocator\_type** = Allocator
- using **value\_type** = std::pair< key\_type, mapped\_type >
- using **container\_type** = std::list< value\_type, allocator\_type >
- using **size\_type** = typename container\_type::size\_type
- using **difference\_type** = typename container\_type::difference\_type
- using **iterator** = typename container\_type::iterator
- using **const\_iterator** = typename container\_type::const\_iterator
- using **reverse\_iterator** = std::reverse\_iterator< iterator >
- using **const\_reverse\_iterator** = std::reverse\_iterator< const\_iterator >
- using **reference** = value\_type &
- using **const\_reference** = const value\_type &
- using **pointer** = typename std::allocator\_traits< allocator\_type >::pointer
- using **const\_pointer** = typename std::allocator\_traits< allocator\_type >::const\_pointer

### Public Member Functions

- **LRUCache** (const size\_t maxSize=100)
- iterator **begin** ()
- const\_iterator **begin** () const
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- iterator **end** ()
- const\_iterator **end** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- bool **empty** () const
- size\_t **size** () const
- size\_t **max\_size** () const
- void **clear** ()
- void **put** (const key\_type &key, const mapped\_type &value)
- const mapped\_type \* **get** (const key\_type &key) const
- bool **exists** (const key\_type &key)
- bool **remove** (const key\_type &key)
- void **setMaxSize** (const size\_t maxSize)

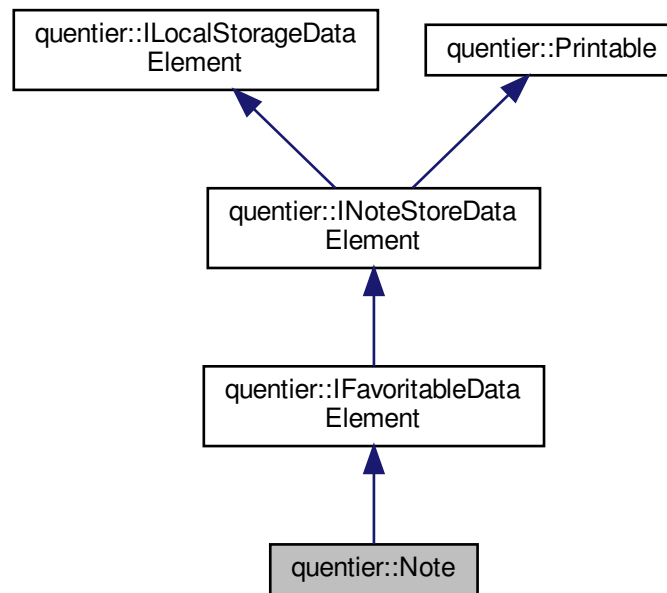
## 4.45 quantier::Note Class Reference

Inheritance diagram for quantier::Note:





Collaboration diagram for `quentier::Note`:



## Public Member Functions

- **Note** (const [Note](#) &other)
- **Note** ([Note](#) &&other)
- [Note](#) & **operator=** (const [Note](#) &other)
- [Note](#) & **operator=** ([Note](#) &&other)
- **Note** (const qevercloud::Note &other)
- [Note](#) & **operator=** (const qevercloud::Note &other)
- bool **operator==** (const [Note](#) &other) const
- bool **operator!=** (const [Note](#) &other) const
- const qevercloud::Note & **qevercloudNote** () const
- qevercloud::Note & **qevercloudNote** ()
- virtual bool **hasGuid** () const override
- virtual const QString & **guid** () const override
- virtual void **setGuid** (const QString &guid) override
- virtual bool **hasUpdateSequenceNumber** () const override
- virtual quint32 **updateSequenceNumber** () const override
- virtual void **setUpdateSequenceNumber** (const quint32 usn) override
- virtual void **clear** () override
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const override
- bool **hasTitle** () const
- const QString & **title** () const
- void **setTitle** (const QString &title)
- bool **hasContent** () const
- const QString & **content** () const
- void **setContent** (const QString &content)



- bool **hasContentHash** () const
- const QByteArray & **contentHash** () const
- void **setContentHash** (const QByteArray &contentHash)
- bool **hasContentLength** () const
- quint32 **contentLength** () const
- void **setContentLength** (const quint32 length)
- bool **hasCreationTimestamp** () const
- quint64 **creationTimestamp** () const
- void **setCreationTimestamp** (const quint64 timestamp)
- bool **hasModificationTimestamp** () const
- quint64 **modificationTimestamp** () const
- void **setModificationTimestamp** (const quint64 timestamp)
- bool **hasDeletionTimestamp** () const
- quint64 **deletionTimestamp** () const
- void **setDeletionTimestamp** (const quint64 timestamp)
- bool **hasActive** () const
- bool **active** () const
- void **setActive** (const bool active)
- bool **hasNotebookGuid** () const
- const QString & **notebookGuid** () const
- void **setNotebookGuid** (const QString &guid)
- bool **hasNotebookLocalUid** () const
- const QString & **notebookLocalUid** () const
- void **setNotebookLocalUid** (const QString &notebookLocalUid)
- bool **hasTagGuids** () const
- const QStringList & **tagGuids** () const
- void **setTagGuids** (const QStringList &guids)
- void **addTagGuid** (const QString &guid)
- void **removeTagGuid** (const QString &guid)
- bool **hasTagLocalUids** () const
- const QStringList & **tagLocalUids** () const
- void **setTagLocalUids** (const QStringList &localUids)
- void **addTagLocalUid** (const QString &localUid)
- void **removeTagLocalUid** (const QString &localUid)
- bool **hasResources** () const
- int **numResources** () const
- QList< [Resource](#) > **resources** () const
- void **setResources** (const QList< [Resource](#) > &resources)
- void **addResource** (const [Resource](#) &resource)
- bool **updateResource** (const [Resource](#) &resource)
- bool **removeResource** (const [Resource](#) &resource)
- bool **hasNoteAttributes** () const
- const qevercloud::NoteAttributes & **noteAttributes** () const
- qevercloud::NoteAttributes & **noteAttributes** ()
- void **clearNoteAttributes** ()
- bool **hasSharedNotes** () const
- QList< [SharedNote](#) > **sharedNotes** () const
- void **setSharedNotes** (const QList< [SharedNote](#) > &sharedNotes)
- void **addSharedNote** (const [SharedNote](#) &sharedNote)
- bool **updateSharedNote** (const [SharedNote](#) &sharedNote)
- bool **removeSharedNote** (const [SharedNote](#) &sharedNote)
- bool **hasNoteRestrictions** () const
- const qevercloud::NoteRestrictions & **noteRestrictions** () const
- qevercloud::NoteRestrictions & **noteRestrictions** ()
- void **setNoteRestrictions** (qevercloud::NoteRestrictions &&restrictions)

- bool **hasNoteLimits** () const
- const qevercloud::NoteLimits & **noteLimits** () const
- qevercloud::NoteLimits & **noteLimits** ()
- void **setNoteLimits** (qevercloud::NoteLimits &&limits)
- QByteArray **thumbnailData** () const
- void **setThumbnailData** (const QByteArray &thumbnailData)
- bool **isInkNote** () const
- QString **plainText** (ErrorString \*pErrorMessage=nullptr) const
- QStringList **listOfWords** (ErrorString \*pErrorMessage=nullptr) const
- std::pair< QString, QStringList > **plainTextAndListOfWords** (ErrorString \*pErrorMessage=nullptr) const
- bool **containsCheckedTodo** () const
- bool **containsUncheckedTodo** () const
- bool **containsTodo** () const
- bool **containsEncryption** () const
- virtual QTextStream & **print** (QTextStream &strm) const override

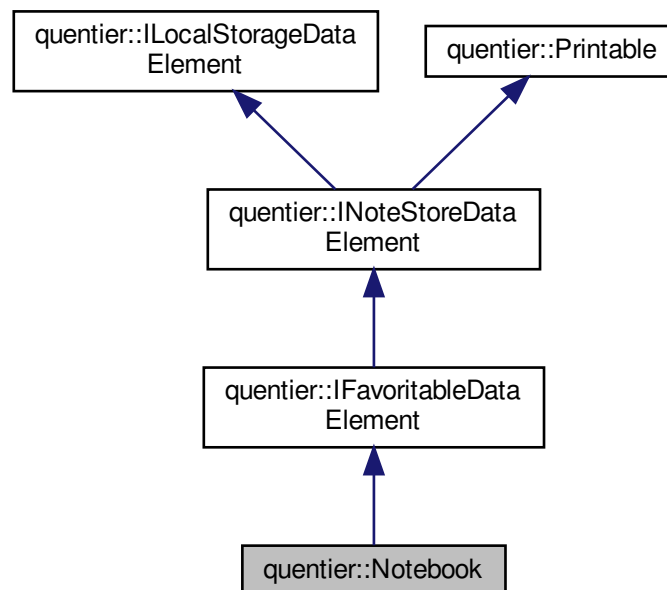
### Static Public Member Functions

- static bool **validateTitle** (const QString &title, ErrorString \*pErrorDescription=nullptr)

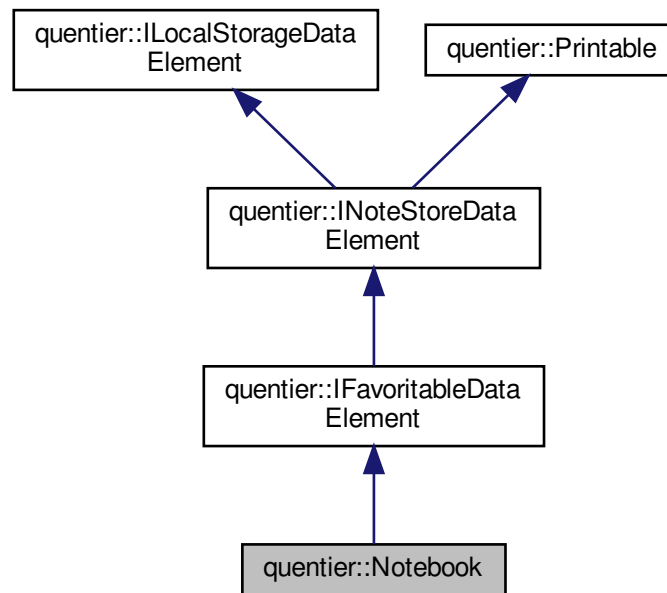
### Additional Inherited Members

## 4.46 quantier::Notebook Class Reference

Inheritance diagram for quantier::Notebook:



Collaboration diagram for quantier::Notebook:



## Public Member Functions

- **Notebook** (const [Notebook](#) &other)
- **Notebook** ([Notebook](#) &&other)
- [Notebook](#) & **operator=** (const [Notebook](#) &other)
- [Notebook](#) & **operator=** ([Notebook](#) &&other)
- **Notebook** (const qevercloud::Notebook &other)
- **Notebook** (qevercloud::Notebook &&other)
- [Notebook](#) & **operator=** (const qevercloud::Notebook &other)
- [Notebook](#) & **operator=** (qevercloud::Notebook &&other)
- bool **operator==** (const [Notebook](#) &other) const
- bool **operator!=** (const [Notebook](#) &other) const
- const qevercloud::Notebook & **qevercloudNotebook** () const
- qevercloud::Notebook & **qevercloudNotebook** ()
- virtual void **clear** () override
- virtual bool **hasGuid** () const override
- virtual const QString & **guid** () const override
- virtual void **setGuid** (const QString &guid) override
- virtual bool **hasUpdateSequenceNumber** () const override
- virtual qint32 **updateSequenceNumber** () const override
- virtual void **setUpdateSequenceNumber** (const qint32 usn) override
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const override
- bool **hasName** () const
- const QString & **name** () const
- void **setName** (const QString &name)
- bool **isDefaultNotebook** () const

- void **setDefaultNotebook** (const bool defaultNotebook)
- bool **hasLinkedNotebookGuid** () const
- const QString & **linkedNotebookGuid** () const
- void **setLinkedNotebookGuid** (const QString &linkedNotebookGuid)
- bool **hasCreationTimestamp** () const
- qint64 **creationTimestamp** () const
- void **setCreationTimestamp** (const qint64 timestamp)
- bool **hasModificationTimestamp** () const
- qint64 **modificationTimestamp** () const
- void **setModificationTimestamp** (const qint64 timestamp)
- bool **hasPublishingUri** () const
- const QString & **publishingUri** () const
- void **setPublishingUri** (const QString &uri)
- bool **hasPublishingOrder** () const
- qint8 **publishingOrder** () const
- void **setPublishingOrder** (const qint8 order)
- bool **hasPublishingAscending** () const
- bool **isPublishingAscending** () const
- void **setPublishingAscending** (const bool ascending)
- bool **hasPublishingPublicDescription** () const
- const QString & **publishingPublicDescription** () const
- void **setPublishingPublicDescription** (const QString &publishingPublicDescription)
- bool **hasPublished** () const
- bool **isPublished** () const
- void **setPublished** (const bool published)
- bool **hasStack** () const
- const QString & **stack** () const
- void **setStack** (const QString &stack)
- bool **hasSharedNotebooks** ()
- QList< [SharedNotebook](#) > **sharedNotebooks** () const
- void **setSharedNotebooks** (QList< qevercloud::SharedNotebook > sharedNotebooks)
- void **setSharedNotebooks** (QList< [SharedNotebook](#) > &&notebooks)
- void **addSharedNotebook** (const [SharedNotebook](#) &sharedNotebook)
- void **removeSharedNotebook** (const [SharedNotebook](#) &sharedNotebook)
- bool **hasBusinessNotebookDescription** () const
- const QString & **businessNotebookDescription** () const
- void **setBusinessNotebookDescription** (const QString &businessNotebookDescription)
- bool **hasBusinessNotebookPrivilegeLevel** () const
- qint8 **businessNotebookPrivilegeLevel** () const
- void **setBusinessNotebookPrivilegeLevel** (const qint8 privilegeLevel)
- bool **hasBusinessNotebookRecommended** () const
- bool **isBusinessNotebookRecommended** () const
- void **setBusinessNotebookRecommended** (const bool recommended)
- bool **hasContact** () const
- const [User](#) **contact** () const
- void **setContact** (const [User](#) &contact)
- bool **isLastUsed** () const
- void **setLastUsed** (const bool lastUsed)
- bool **canReadNotes** () const
- void **setCanReadNotes** (const bool canReadNotes)
- bool **canCreateNotes** () const
- void **setCanCreateNotes** (const bool canCreateNotes)
- bool **canUpdateNotes** () const
- void **setCanUpdateNotes** (const bool canUpdateNotes)
- bool **canExpungeNotes** () const

- void **setCanExpungeNotes** (const bool canExpungeNotes)
- bool **canShareNotes** () const
- void **setCanShareNotes** (const bool canShareNotes)
- bool **canEmailNotes** () const
- void **setCanEmailNotes** (const bool canEmailNotes)
- bool **canSendMessageToRecipients** () const
- void **setCanSendMessageToRecipients** (const bool canSendMessageToRecipients)
- bool **canUpdateNotebook** () const
- void **setCanUpdateNotebook** (const bool canUpdateNotebook)
- bool **canExpungeNotebook** () const
- void **setCanExpungeNotebook** (const bool canExpungeNotebook)
- bool **canSetDefaultNotebook** () const
- void **setCanSetDefaultNotebook** (const bool canSetDefaultNotebook)
- bool **canSetNotebookStack** () const
- void **setCanSetNotebookStack** (const bool canSetNotebookStack)
- bool **canPublishToPublic** () const
- void **setCanPublishToPublic** (const bool canPublishToPublic)
- bool **canPublishToBusinessLibrary** () const
- void **setCanPublishToBusinessLibrary** (const bool canPublishToBusinessLibrary)
- bool **canCreateTags** () const
- void **setCanCreateTags** (const bool canCreateTags)
- bool **canUpdateTags** () const
- void **setCanUpdateTags** (const bool canUpdateTags)
- bool **canExpungeTags** () const
- void **setCanExpungeTags** (const bool canExpungeTags)
- bool **canSetParentTag** () const
- void **setCanSetParentTag** (const bool canSetParentTag)
- bool **canCreateSharedNotebooks** () const
- void **setCanCreateSharedNotebooks** (const bool canCreateSharedNotebooks)
- bool **canShareNotesWithBusiness** () const
- void **setCanShareNotesWithBusiness** (const bool canShareNotesWithBusiness)
- bool **canRenameNotebook** () const
- void **setCanRenameNotebook** (const bool canRenameNotebook)
- bool **hasUpdateWhichSharedNotebookRestrictions** () const
- qint8 **updateWhichSharedNotebookRestrictions** () const
- void **setUpdateWhichSharedNotebookRestrictions** (const qint8 which)
- bool **hasExpungeWhichSharedNotebookRestrictions** () const
- qint8 **expungeWhichSharedNotebookRestrictions** () const
- void **setExpungeWhichSharedNotebookRestrictions** (const qint8 which)
- bool **hasRestrictions** () const
- const qevercloud::NotebookRestrictions & **restrictions** () const
- void **setNotebookRestrictions** (qevercloud::NotebookRestrictions &&restrictions)
- bool **hasRecipientReminderNotifyEmail** () const
- bool **recipientReminderNotifyEmail** () const
- void **setRecipientReminderNotifyEmail** (const bool notifyEmail)
- bool **hasRecipientReminderNotifyInApp** () const
- bool **recipientReminderNotifyInApp** () const
- void **setRecipientReminderNotifyInApp** (const bool notifyInApp)
- bool **hasRecipientInMyList** () const
- bool **recipientInMyList** () const
- void **setRecipientInMyList** (const bool inMyList)
- bool **hasRecipientStack** () const
- const QString & **recipientStack** () const
- void **setRecipientStack** (const QString &recipientString)
- bool **hasRecipientSettings** () const
- const qevercloud::NotebookRecipientSettings & **recipientSettings** () const
- void **setNotebookRecipientSettings** (qevercloud::NotebookRecipientSettings &&settings)
- virtual QTextStream & **print** (QTextStream &strm) const override

## Static Public Member Functions

- static bool **validateName** (const QString &name, [ErrorString](#) \*pErrorDescription=nullptr)

## Additional Inherited Members

## 4.47 quantier::ENMLConverter::NoteContentToHtmlExtraData Struct Reference

### Public Attributes

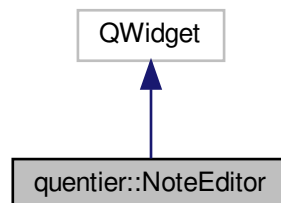
- quint64 **m\_numEnToDoNodes** = 0
- quint64 **m\_numHyperlinkNodes** = 0
- quint64 **m\_numEnCryptNodes** = 0
- quint64 **m\_numEnDecryptedNodes** = 0

## 4.48 quantier::NoteEditor Class Reference

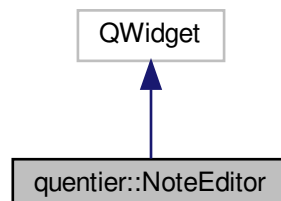
The [NoteEditor](#) class is a widget encapsulating all the functionality necessary for showing and editing notes.

```
#include <NoteEditor.h>
```

Inheritance diagram for quantier::NoteEditor:



Collaboration diagram for quantier::NoteEditor:



## Public Slots

- void [convertToNote](#) ()
- void [saveNoteToLocalStorage](#) ()
- void [setNoteTitle](#) (const QString &noteTitle)
- void [setTagIds](#) (const QStringList &tagLocalUids, const QStringList &tagGuids)
- void **undo** ()
- void **redo** ()
- void **cut** ()
- void **copy** ()
- void **paste** ()
- void **pasteUnformatted** ()
- void **selectAll** ()
- void **formatSelectionAsSourceCode** ()
- void **fontMenu** ()
- void **textBold** ()
- void **textItalic** ()
- void **textUnderline** ()
- void **textStrikethrough** ()
- void **textHighlight** ()
- void **alignLeft** ()
- void **alignCenter** ()
- void **alignRight** ()
- void **alignFull** ()
- void **findNext** (const QString &text, const bool matchCase) const
- void **findPrevious** (const QString &text, const bool matchCase) const
- void **replace** (const QString &textToReplace, const QString &replacementText, const bool matchCase)
- void **replaceAll** (const QString &textToReplace, const QString &replacementText, const bool matchCase)
- void **insertToDoCheckbox** ()
- void **insertInAppNoteLink** (const QString &userId, const QString &shardId, const QString &noteGuid, const QString &linkText)
- void **setSpellcheck** (const bool enabled)
- void **setFont** (const QFont &font)
- void **setFontHeight** (const int height)
- void **setFontColor** (const QColor &color)
- void **setBackgroundColor** (const QColor &color)
- void [setDefaultPalette](#) (const QPalette &pal)
- void [setDefaultFont](#) (const QFont &font)
- void **insertHorizontalLine** ()
- void **increaseFontSize** ()
- void **decreaseFontSize** ()
- void **increaseIndentation** ()
- void **decreaseIndentation** ()
- void **insertBulletedList** ()
- void **insertNumberedList** ()
- void **insertTableDialog** ()
- void **insertFixedWidthTable** (const int rows, const int columns, const int widthInPixels)
- void **insertRelativeWidthTable** (const int rows, const int columns, const double relativeWidth)
- void **insertTableRow** ()
- void **insertTableColumn** ()
- void **removeTableRow** ()
- void **removeTableColumn** ()
- void **addAttachmentDialog** ()
- void **saveAttachmentDialog** (const QByteArray &resourceHash)
- void **saveAttachmentUnderCursor** ()

- void **openAttachment** (const QByteArray &resourceHash)
- void **openAttachmentUnderCursor** ()
- void **copyAttachment** (const QByteArray &resourceHash)
- void **copyAttachmentUnderCursor** ()
- void **encryptSelectedText** ()
- void **decryptEncryptedTextUnderCursor** ()
- void **editHyperlinkDialog** ()
- void **copyHyperlink** ()
- void **removeHyperlink** ()
- void **onNoteLoadCancelled** ()

## Signals

- void **contentChanged** ()  
*contentChanged signal is emitted when the note's content (text) gets modified via manual editing (i.e. not any action like paste or cut)*
- void **noteAndNotebookFoundInLocalStorage** ([Note](#) note, [Notebook](#) notebook)  
*noteAndNotebookFoundInLocalStorage signal is emitted when note and its corresponding notebook were found within the local storage right before the note editor starts to load the note into the editor*
- void **noteNotFound** (QString noteLocalUid)  
*noteNotFound signal is emitted when the note could not be found within the local storage by the provided local uid*
- void **noteDeleted** (QString noteLocalUid)  
*noteDeleted signal is emitted when the note displayed within the note editor is deleted. The note editor stops displaying the note in this case shortly after emitting this signal*
- void **noteModified** ()  
*noteModified signal is emitted when the note's content within the editor gets modified via some way - either via manual editing or via some action (like paste or cut)*
- void **notifyError** ([ErrorString](#) error)  
*notifyError signal is emitted when [NoteEditor](#) encounters some problem worth letting the user to know about*
- void **inAppNoteLinkClicked** (QString userId, QString shardId, QString noteGuid)  
*inAppNoteLinkClicked signal is emitted when the in-app note link is clicked within the note editor*
- void **inAppNoteLinkPasteRequested** (QString url, QString userId, QString shardId, QString noteGuid)
- void **convertedToNote** ([Note](#) note)
- void **cantConvertToNote** ([ErrorString](#) error)
- void **noteEditorHtmlUpdated** (QString html)
- void **currentNoteChanged** ([Note](#) note)
- void **spellCheckerNotReady** ()
- void **spellCheckerReady** ()
- void **noteLoaded** ()
- void **noteSavedToLocalStorage** (QString noteLocalUid)  
*noteSavedToLocalStorage signal is emitted when the note has been saved within the local storage. [NoteEditor](#) doesn't do this on its own unless it's explicitly asked to do this via invoking its `saveNoteToLocalStorage` slot*
- void **failedToSaveNoteToLocalStorage** ([ErrorString](#) errorDescription, QString noteLocalUid)  
*failedToSaveNoteToLocalStorage signal is emitted in case of failure to save the note to local storage*
- void **textBoldState** (bool state)
- void **textItalicState** (bool state)
- void **textUnderlineState** (bool state)
- void **textStrikethroughState** (bool state)
- void **textAlignLeftState** (bool state)
- void **textAlignCenterState** (bool state)
- void **textAlignRightState** (bool state)
- void **textAlignFullState** (bool state)
- void **textInsideOrderedListState** (bool state)



- void **textInsideUnorderedListState** (bool state)
- void **textInsideTableState** (bool state)
- void **textFontFamilyChanged** (QString fontFamily)
- void **textFontSizeChanged** (int fontSize)
- void **insertTableDialogRequested** ()

## Public Member Functions

- **NoteEditor** (QWidget \*parent=nullptr, Qt::WindowFlags flags={})
- void **initialize** (LocalStorageManagerAsync &localStorageManager, SpellChecker &spellChecker, const Account &account, QThread \*pBackgroundJobsThread=nullptr)
- INoteEditorBackend \* **backend** ()
- void **setBackend** (INoteEditorBackend \*backend)
- void **setAccount** (const Account &account)
- const QUndoStack \* **undoStack** () const
- void **setUndoStack** (QUndoStack \*pUndoStack)
- void **setInitialPageHtml** (const QString &html)
- void **setNoteNotFoundPageHtml** (const QString &html)
- void **setNoteDeletedPageHtml** (const QString &html)
- void **setNoteLoadingPageHtml** (const QString &html)
- QString **currentNoteLocalUid** () const
- void **setCurrentNoteLocalUid** (const QString &noteLocalUid)
- void **clear** ()
- bool **isModified** () const
- bool **isEditorPageModified** () const
- bool **isNoteLoaded** () const
- qint64 **idleTime** () const
- void **setFocus** ()
- QString **selectedText** () const
- bool **hasSelection** () const
- bool **spellCheckEnabled** () const
- bool **print** (QPrinter &printer, ErrorString &errorDescription)
- bool **exportToPdf** (const QString &absoluteFilePath, ErrorString &errorDescription)
- bool **exportToEnex** (const QStringList &tagNames, QString &enex, ErrorString &errorDescription)
- QPalette **defaultPalette** () const
- const QFont \* **defaultFont** () const

## Protected Member Functions

- virtual void **dragMoveEvent** (QDragMoveEvent \*pEvent) override
- virtual void **dropEvent** (QDropEvent \*pEvent) override

### 4.48.1 Detailed Description

The `NoteEditor` class is a widget encapsulating all the functionality necessary for showing and editing notes.

### 4.48.2 Member Function Documentation

#### 4.48.2.1 backend()

```
INoteEditorBackend* quantier::NoteEditor::backend ( )
```

##### Returns

the pointer to the note editor's backend

#### 4.48.2.2 clear()

```
void quantier::NoteEditor::clear ( )
```

Clear the contents of the note editor

#### 4.48.2.3 convertToNote

```
void quantier::NoteEditor::convertToNote ( ) [slot]
```

Invoke this slot to launch the asynchronous procedure of converting the current contents of the note editor to note; the convertedToNote signal would be emitted in response when the conversion is done

#### 4.48.2.4 currentNoteLocalUid()

```
QString quantier::NoteEditor::currentNoteLocalUid ( ) const
```

Get the local uid of the note currently set to the note editor

#### 4.48.2.5 defaultFont()

```
const QFont* quantier::NoteEditor::defaultFont ( ) const
```

##### Returns

pointer to the default font used by the note editor; if no such font was set to the editor previously, returns null pointer

#### 4.48.2.6 defaultPalette()

```
QPalette quantier::NoteEditor::defaultPalette ( ) const
```

##### Returns

palette containing default colors used by the editor; the palette is composed of colors from note editor widget's native palette but some of them might be overridden by colors from the palette specified previously via set↔DefaultPalette method: those colors from the specified palette which were valid

#### 4.48.2.7 `idleTime()`

```
qint64 quentier::NoteEditor::idleTime ( ) const
```

##### Returns

the number of milliseconds since the last user's interaction with the note editor or -1 if there was no interaction or if no note is loaded at the moment

#### 4.48.2.8 `inAppNoteLinkPasteRequested`

```
void quentier::NoteEditor::inAppNoteLinkPasteRequested (
    QString url,
    QString userId,
    QString shardId,
    QString noteGuid ) [signal]
```

`inAppNoteLinkPasteRequested` signal is emitted when the note editor detects the attempt to paste the in-app note link into the note editor; the link would not be inserted right away, instead this signal would be emitted. Whatever party managing the note editor is expected to connect some slot to this signal and provide the optionally amended link information to the note editor by sending the signal connected to its `insertInAppNoteLink` slot - this slot accepts both the URL of the link and the link text and performs the actual link insertion into the note. If the link text is empty, the URL itself is used as the link text.

#### 4.48.2.9 `initialize()`

```
void quentier::NoteEditor::initialize (
    LocalStorageManagerAsync & localStorageManager,
    SpellChecker & spellChecker,
    const Account & account,
    QThread * pBackgroundJobsThread = nullptr )
```

`NoteEditor` requires `LocalStorageManagerAsync`, `SpellChecker` and `Account` for its work but due to the particularities of Qt's .ui files processing these can't be passed right inside the constructor, hence here's a special initialization method

##### Parameters

<i>localStorageManager</i>	The reference to <code>LocalStorageManagerAsync</code> , to set up signal-slot connections with it
<i>spellChecker</i>	The spell checker to be used by note editor for, well, spell-checking
<i>account</i>	Currently active account
<i>pBackgroundJobsThread</i>	Pointer to the thread to be used for scheduling of background jobs of <code>NoteEditor</code> ; if null, <code>NoteEditor</code> 's background jobs would take place in GUI thread

#### 4.48.2.10 isEditorPageModified()

```
bool quantier::NoteEditor::isEditorPageModified ( ) const
```

##### Returns

true if there's content within the editor not yet converted to note, false otherwise

#### 4.48.2.11 isModified()

```
bool quantier::NoteEditor::isModified ( ) const
```

##### Returns

true if there's content within the editor not yet converted to note or not saved to local storage, false otherwise

#### 4.48.2.12 isNoteLoaded()

```
bool quantier::NoteEditor::isNoteLoaded ( ) const
```

##### Returns

true if the note last set to the editor has been fully loaded already, false otherwise

#### 4.48.2.13 saveNoteToLocalStorage

```
void quantier::NoteEditor::saveNoteToLocalStorage ( ) [slot]
```

Invoke this slot to launch the asynchronous procedure of saving the modified current note back to the local storage. If no note is set to the editor or if the note is not modified, no action would be performed. Otherwise `noteSavedToLocalStorage` signal would be emitted in case of successful saving or `failedToSaveNoteToLocalStorage` would be emitted otherwise

#### 4.48.2.14 setAccount()

```
void quantier::NoteEditor::setAccount (
    const Account & account )
```

Set the current account to the note editor

#### 4.48.2.15 setBackend()

```
void quantier::NoteEditor::setBackend (
    INoteEditorBackend * backend )
```

This method can be used to set the backend to the note editor; the note editor has the default backend so this method is not obligatory to be called

#### 4.48.2.16 setCurrentNoteLocalUid()

```
void quantier::NoteEditor::setCurrentNoteLocalUid (
    const QString & noteLocalUid )
```

Set note local uid to the note editor. The note is being searched for within the local storage, in case of no note being found `noteNotFound` signal is emitted. Otherwise note editor page starts loading.

## Parameters

<i>noteLocalUid</i>	The local uid of note
---------------------	-----------------------

**4.48.2.17 setDefaultFont**

```
void quantier::NoteEditor::setDefaultFont (
    const QFont & font ) [slot]
```

Sets the font which would be used by the editor by default

## Parameters

<i>font</i>	The font to be used by the editor by default
-------------	--

**4.48.2.18 setDefaultPalette**

```
void quantier::NoteEditor::setDefaultPalette (
    const QPalette & pal ) [slot]
```

Sets the palette with colors to be used by the editor. New colors are applied after the note is fully loaded. If no note is set to the editor, the palette is simply remembered for the next note to be loaded into it.

Colors within the palette and their usage:

1. WindowText - used as default font color
2. Base - used as default background color
3. HighlightedText - used as font color for selected text
4. Highlight - used as background color for selected text

## Parameters

<i>pal</i>	The palette to be set. Invalid colors from it are substituted by colors from widget's palette by the editor
------------	---

**4.48.2.19 setFocus()**

```
void quantier::NoteEditor::setFocus ( )
```

Sets the focus to the backend note editor widget

#### 4.48.2.20 setInitialPageHtml()

```
void quentier::NoteEditor::setInitialPageHtml (
    const QString & html )
```

Set the html to be displayed when the note is not set to the editor

#### 4.48.2.21 setNoteDeletedPageHtml()

```
void quentier::NoteEditor::setNoteDeletedPageHtml (
    const QString & html )
```

Set the html to be displayed when the note set to the editor was deleted from the local storage (either marked as deleted or deleted permanently i.e. expunged)

#### 4.48.2.22 setNoteLoadingPageHtml()

```
void quentier::NoteEditor::setNoteLoadingPageHtml (
    const QString & html )
```

Set the html to be displayed when the note set to the editor is being loaded into it

#### 4.48.2.23 setNoteNotFoundPageHtml()

```
void quentier::NoteEditor::setNoteNotFoundPageHtml (
    const QString & html )
```

Set the html to be displayed when the note attempted to be set to the editor was not found within the local storage

#### 4.48.2.24 setNoteTitle

```
void quentier::NoteEditor::setNoteTitle (
    const QString & noteTitle ) [slot]
```

Invoke this slot to set the title to the note displayed via the note editor. The note editor itself doesn't manage the note title in any way so any external code using the note editor can set the title to the note editor's note which would be considered modified if the title is new and then eventually the note would be saved to local storage

##### Parameters

<i>noteTitle</i>	The title of the note
------------------	-----------------------

#### 4.48.2.25 setTagIds

```
void quentier::NoteEditor::setTagIds (
```

```
const QStringList & tagLocalUids,
const QStringList & tagGuids ) [slot]
```

Invoke this slot to set tag local uids and/or tag guides to the note displayed via the note editor. The note editor itself doesn't manage the note tags in any way so any external code using the note editor can set the tag ids to the note editor's internal note which would be considered modified if the tag ids are new and then eventually the note would be saved to local storage

#### Parameters

<i>tagLocalUids</i>	The list of tag local uids for the note
<i>tagGuids</i>	The list of tag guides for the note

#### 4.48.2.26 setUndoStack()

```
void quantier::NoteEditor::setUndoStack (
    QUndoStack * pUndoStack )
```

Set the undo stack for the note editor to use

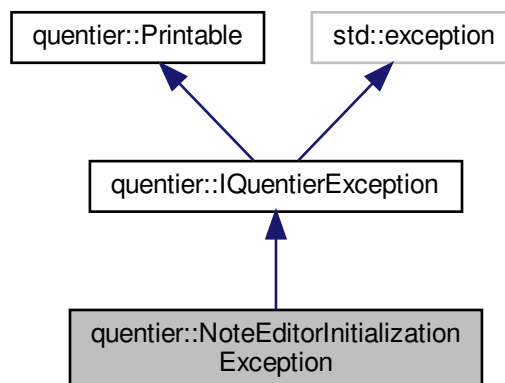
#### 4.48.2.27 undoStack()

```
const QUndoStack* quantier::NoteEditor::undoStack ( ) const
```

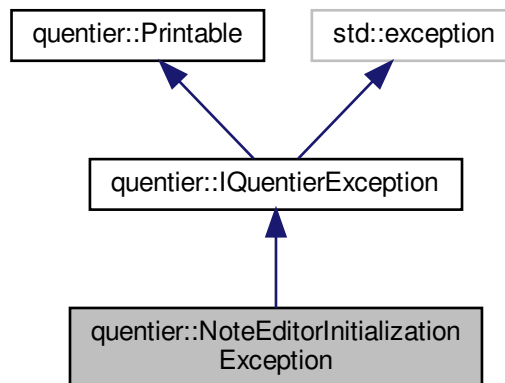
Get the undo stack serving to the note editor

## 4.49 quantier::NoteEditorInitializationException Class Reference

Inheritance diagram for quantier::NoteEditorInitializationException:



Collaboration diagram for `quentier::NoteEditorInitializationException`:



### Public Member Functions

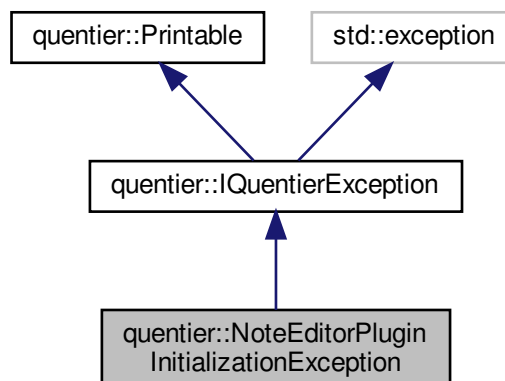
- **NoteEditorInitializationException** (const [ErrorString](#) &message)

### Protected Member Functions

- virtual const QString **exceptionDisplayName** () const override

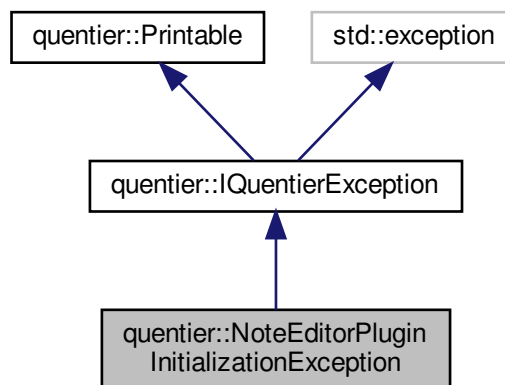
## 4.50 quentier::NoteEditorPluginInitializationException Class Reference

Inheritance diagram for `quentier::NoteEditorPluginInitializationException`:





Collaboration diagram for quantier::NoteEditorPluginInitializationException:



### Public Member Functions

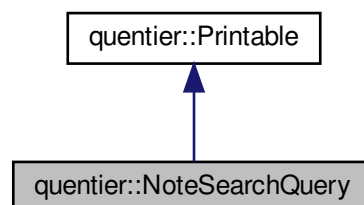
- **NoteEditorPluginInitializationException** (const [ErrorString](#) &message)

### Protected Member Functions

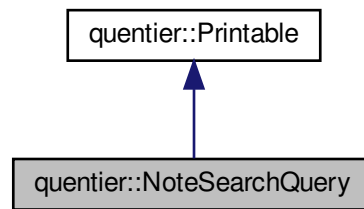
- virtual const QString **exceptionDisplayName** () const override

## 4.51 quantier::NoteSearchQuery Class Reference

Inheritance diagram for quantier::NoteSearchQuery:



Collaboration diagram for `quentier::NoteSearchQuery`:



## Public Member Functions

- **NoteSearchQuery** (const [NoteSearchQuery](#) &other)
- **NoteSearchQuery** ([NoteSearchQuery](#) &&other)
- [NoteSearchQuery](#) & **operator=** (const [NoteSearchQuery](#) &other)
- [NoteSearchQuery](#) & **operator=** ([NoteSearchQuery](#) &&other)
- bool **isEmpty** () const
- void **clear** ()
- const QString **queryString** () const
- bool **setQueryString** (const QString &[queryString](#), [ErrorString](#) &error)
- const QString **notebookModifier** () const
- bool **hasAnyModifier** () const
- const QStringList & **tagNames** () const
- const QStringList & **negatedTagNames** () const
- bool **hasAnyTag** () const
- bool **hasNegatedAnyTag** () const
- const QStringList & **titleNames** () const
- const QStringList & **negatedTitleNames** () const
- bool **hasAnyTitleName** () const
- bool **hasNegatedAnyTitleName** () const
- const QVector< qint64 > & **creationTimestamps** () const
- const QVector< qint64 > & **negatedCreationTimestamps** () const
- bool **hasAnyCreationTimestamp** () const
- bool **hasNegatedAnyCreationTimestamp** () const
- const QVector< qint64 > & **modificationTimestamps** () const
- const QVector< qint64 > & **negatedModificationTimestamps** () const
- bool **hasAnyModificationTimestamp** () const
- bool **hasNegatedAnyModificationTimestamp** () const
- const QStringList & **resourceMimeTypes** () const
- const QStringList & **negatedResourceMimeTypes** () const
- bool **hasAnyResourceMimeType** () const
- bool **hasNegatedAnyResourceMimeType** () const
- const QVector< qint64 > & **subjectDateTimestamps** () const
- const QVector< qint64 > & **negatedSubjectDateTimestamps** () const
- bool **hasAnySubjectDateTimestamp** () const
- bool **hasNegatedAnySubjectDateTimestamp** () const
- const QVector< double > & **latitudes** () const

- const QVector< double > & **negatedLatitudes** () const
- bool **hasAnyLatitude** () const
- bool **hasNegatedAnyLatitude** () const
- const QVector< double > & **longitudes** () const
- const QVector< double > & **negatedLongitudes** () const
- bool **hasAnyLongitude** () const
- bool **hasNegatedAnyLongitude** () const
- const QVector< double > & **altitudes** () const
- const QVector< double > & **negatedAltitudes** () const
- bool **hasAnyAltitude** () const
- bool **hasNegatedAnyAltitude** () const
- const QStringList & **authors** () const
- const QStringList & **negatedAuthors** () const
- bool **hasAnyAuthor** () const
- bool **hasNegatedAnyAuthor** () const
- const QStringList & **sources** () const
- const QStringList & **negatedSources** () const
- bool **hasAnySource** () const
- bool **hasNegatedAnySource** () const
- const QStringList & **sourceApplications** () const
- const QStringList & **negatedSourceApplications** () const
- bool **hasAnySourceApplication** () const
- bool **hasNegatedAnySourceApplication** () const
- const QStringList & **contentClasses** () const
- const QStringList & **negatedContentClasses** () const
- bool **hasAnyContentClass** () const
- bool **hasNegatedAnyContentClass** () const
- const QStringList & **placeNames** () const
- const QStringList & **negatedPlaceNames** () const
- bool **hasAnyPlaceName** () const
- bool **hasNegatedAnyPlaceName** () const
- const QStringList & **applicationData** () const
- const QStringList & **negatedApplicationData** () const
- bool **hasAnyApplicationData** () const
- bool **hasNegatedAnyApplicationData** () const
- const QVector< qint64 > & **reminderOrders** () const
- const QVector< qint64 > & **negatedReminderOrders** () const
- bool **hasAnyReminderOrder** () const
- bool **hasNegatedAnyReminderOrder** () const
- const QVector< qint64 > & **reminderTimes** () const
- const QVector< qint64 > & **negatedReminderTimes** () const
- bool **hasAnyReminderTime** () const
- bool **hasNegatedAnyReminderTime** () const
- const QVector< qint64 > & **reminderDoneTimes** () const
- const QVector< qint64 > & **negatedReminderDoneTimes** () const
- bool **hasAnyReminderDoneTime** () const
- bool **hasNegatedAnyReminderDoneTime** () const
- bool **hasUnfinishedToDo** () const
- bool **hasNegatedUnfinishedToDo** () const
- bool **hasFinishedToDo** () const
- bool **hasNegatedFinishedToDo** () const
- bool **hasAnyToDo** () const
- bool **hasNegatedAnyToDo** () const
- bool **hasEncryption** () const
- bool **hasNegatedEncryption** () const

- `const QStringList & contentSearchTerms () const`
- `const QStringList & negatedContentSearchTerms () const`
- `bool hasAnyContentSearchTerms () const`
- `bool isMatcheable () const`
- `virtual QTextStream & print (QTextStream &strm) const` override

## Additional Inherited Members

### 4.51.1 Member Function Documentation

#### 4.51.1.1 `notebookModifier()`

```
const QString quentier::NoteSearchQuery::notebookModifier ( ) const
```

If query string has "notebook:<notebook name>" scope modifier, this method returns the name of the notebook, otherwise it returns empty string

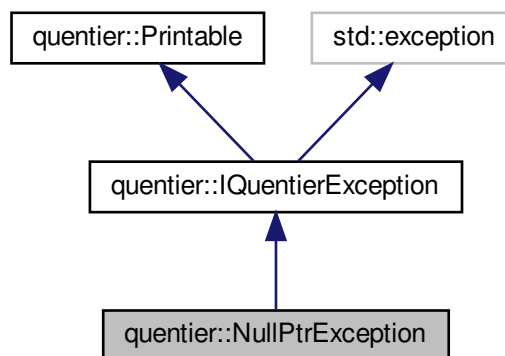
#### 4.51.1.2 `queryString()`

```
const QString quentier::NoteSearchQuery::queryString ( ) const
```

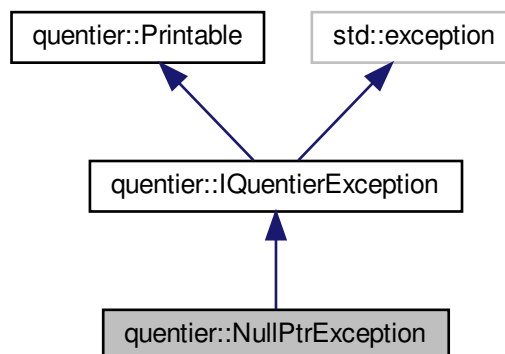
Returns the original non-parsed query string

## 4.52 `quentier::NullPtrException` Class Reference

Inheritance diagram for `quentier::NullPtrException`:



Collaboration diagram for quantier::NullPtrException:



### Public Member Functions

- `NullPtrException` (const [ErrorString](#) &message)

### Protected Member Functions

- virtual const `QString` `exceptionDisplayName` () const override

## 4.53 quantier::ResourceRecognitionIndexItem::ObjectItem Struct Reference

### Public Member Functions

- bool `operator==` (const [ObjectItem](#) &other) const

### Public Attributes

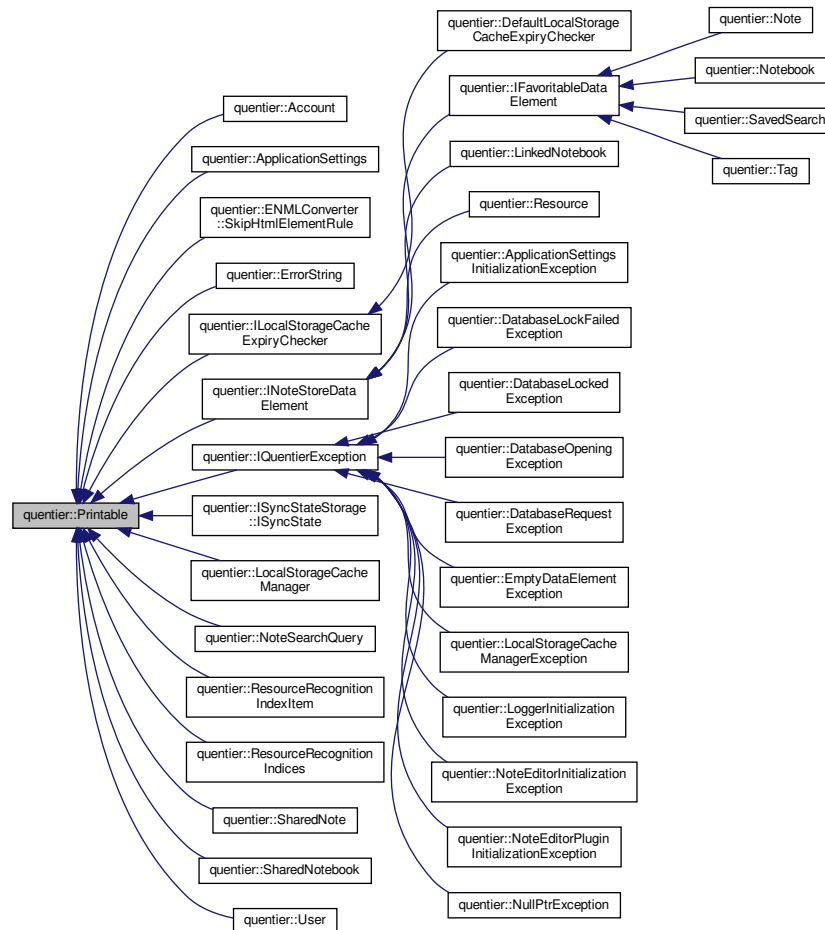
- `QString` `m_objectType`
- int `m_weight` = -1

## 4.54 quantier::Printable Class Reference

The [Printable](#) class is the interface for Quantier's internal classes which should be able to write themselves into QTextStream and/or convert to QString.

```
#include <Printable.h>
```

Inheritance diagram for quantier::Printable:



### Public Member Functions

- virtual QTextStream & **print** (QTextStream &strm) const =0
- virtual const QString **toString** () const

### Protected Member Functions

- **Printable** (const [Printable](#) &other)
- [Printable](#) & **operator=** (const [Printable](#) &other)

## Friends

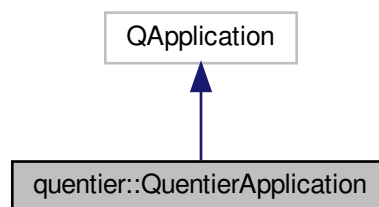
- QUENTIER\_EXPORT QTextStream & **operator**<< (QTextStream &strm, const Printable &printable)
- QUENTIER\_EXPORT QDebug & **operator**<< (QDebug &debug, const Printable &printable)

### 4.54.1 Detailed Description

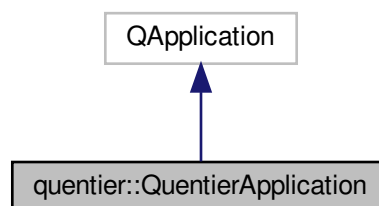
The [Printable](#) class is the interface for Quantier's internal classes which should be able to write themselves into QTextStream and/or convert to QString.

## 4.55 quantier::QuantierApplication Class Reference

Inheritance diagram for quantier::QuantierApplication:



Collaboration diagram for quantier::QuantierApplication:



## Public Member Functions

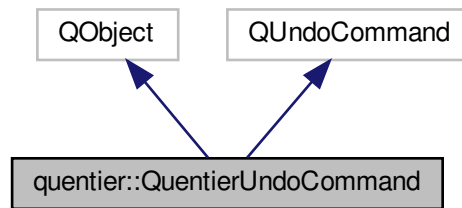
- **QuantierApplication** (int &argc, char \*argv[])
- virtual bool **notify** (QObject \*pObject, QEvent \*pEvent) override
- virtual bool **event** (QEvent \*pEvent) override

## 4.56 quantier::QuantierUndoCommand Class Reference

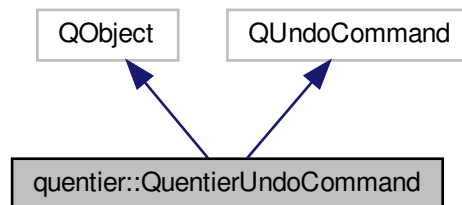
The [QuantierUndoCommand](#) class has the sole purpose of working around one quirky aspect of Qt's undo/redo framework: when you push `QUndoCommand` to `QUndoStack`, it calls "redo" method of that command. This class offers subclasses to implement their own methods for actual "undo" and "redo" commands while ignoring the attempts to "redo" anything if there were no previous "undo" call prior to that.

```
#include <QuantierUndoCommand.h>
```

Inheritance diagram for `quantier::QuantierUndoCommand`:



Collaboration diagram for `quantier::QuantierUndoCommand`:



### Signals

- void **notifyError** ([ErrorString](#) error)

### Public Member Functions

- **QuantierUndoCommand** (`QUndoCommand *parent=nullptr`)
- **QuantierUndoCommand** (`const QString &text, QUndoCommand *parent=nullptr`)
- virtual void **undo** () override final
- virtual void **redo** () override final
- bool **onceUndoExecuted** () const



## Protected Member Functions

- virtual void **undoImpl** ()=0
- virtual void **redoImpl** ()=0

### 4.56.1 Detailed Description

The [QuantierUndoCommand](#) class has the sole purpose of working around one quirky aspect of Qt's undo/redo framework: when you push QUndoCommand to QUndoStack, it calls "redo" method of that command. This class offers subclasses to implement their own methods for actual "undo" and "redo" commands while ignoring the attempts to "redo" anything if there were no previous "undo" call prior to that.

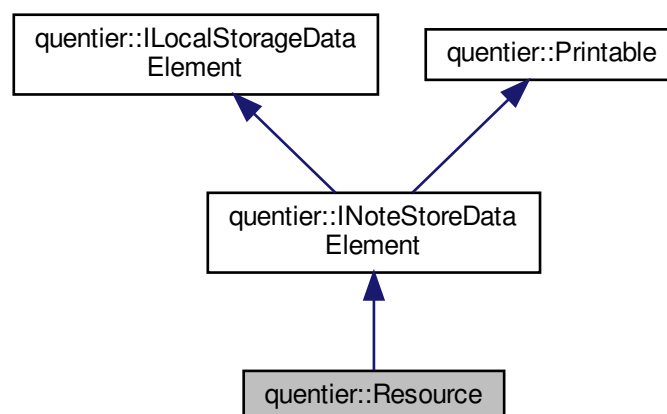
The rationale behind the current behaviour seems to be the compliance with "command pattern behaviour" when you create the command to execute the action instead of just executing it immediately. This design is enforced by Qt's undo/redo framework, there's no option to choose not to call "redo" when pushing to the stack.

One thing which this design fails to see is the fact that the command may be already executed externally by the moment the QUndoCommand can be created. Suppose we can get the information about how to undo (and then again redo) that command. We create the corresponding QUndoCommand, set up the stuff for its undo/redo methods and push it to QUndoStack for future use... But at the same time QUndoStack calls "redo" method of the command. Really not the behaviour you'd like to have.

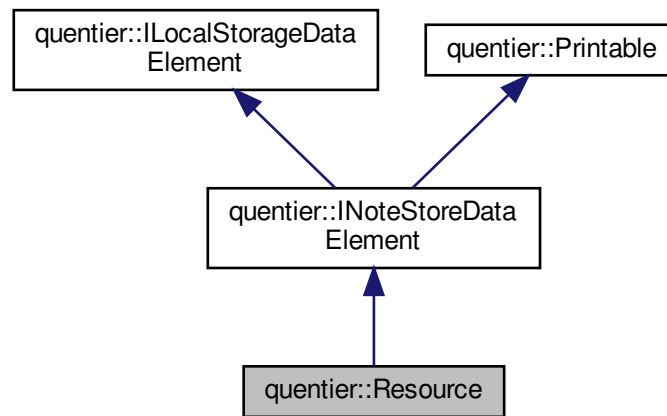
[QuantierUndoCommand](#) is also QObject, it is for error reporting via notifyError signal

## 4.57 quantier::Resource Class Reference

Inheritance diagram for quantier::Resource:



Collaboration diagram for `quentier::Resource`:



## Public Member Functions

- **Resource** (const [Resource](#) &other)
- **Resource** ([Resource](#) &&other)
- **Resource** (const `qevercloud::Resource` &resource)
- [Resource](#) & **operator=** (const [Resource](#) &other)
- [Resource](#) & **operator=** ([Resource](#) &&other)
- bool **operator==** (const [Resource](#) &other) const
- bool **operator!=** (const [Resource](#) &other) const
- const `qevercloud::Resource` & **qevercloudResource** () const
- `qevercloud::Resource` & **qevercloudResource** ()
- virtual void **clear** () override
- virtual bool **hasGuid** () const override
- virtual const `QString` & **guid** () const override
- virtual void **setGuid** (const `QString` &guid) override
- virtual bool **hasUpdateSequenceNumber** () const override
- virtual `qint32` **updateSequenceNumber** () const override
- virtual void **setUpdateSequenceNumber** (const `qint32` updateSequenceNumber) override
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const override
- `QString` **displayName** () const
- void **setDisplayNames** (const `QString` &displayName)
- `QString` **preferredFileSuffix** () const
- int **indexInNote** () const
- void **setIndexInNote** (const int index)
- bool **hasNoteGuid** () const
- const `QString` & **noteGuid** () const
- void **setNoteGuid** (const `QString` &guid)
- bool **hasNoteLocalUuid** () const
- const `QString` & **noteLocalUuid** () const
- void **setNoteLocalUuid** (const `QString` &guid)
- bool **hasData** () const

- `bool hasDataHash () const`
- `const QByteArray & dataHash () const`
- `void setDataHash (const QByteArray &hash)`
- `bool hasDataSize () const`
- `qint32 dataSize () const`
- `void setDataSize (const qint32 size)`
- `bool hasDataBody () const`
- `const QByteArray & dataBody () const`
- `void setDataBody (const QByteArray &body)`
- `bool hasMime () const`
- `const QString & mime () const`
- `void setMime (const QString &mime)`
- `bool hasWidth () const`
- `qint16 width () const`
- `void setWidth (const qint16 width)`
- `bool hasHeight () const`
- `qint16 height () const`
- `void setHeight (const qint16 height)`
- `bool hasRecognitionData () const`
- `bool hasRecognitionDataHash () const`
- `const QByteArray & recognitionDataHash () const`
- `void setRecognitionDataHash (const QByteArray &hash)`
- `bool hasRecognitionDataSize () const`
- `qint32 recognitionDataSize () const`
- `void setRecognitionDataSize (const qint32 size)`
- `bool hasRecognitionDataBody () const`
- `const QByteArray & recognitionDataBody () const`
- `void setRecognitionDataBody (const QByteArray &body)`
- `bool hasAlternateData () const`
- `bool hasAlternateDataHash () const`
- `const QByteArray & alternateDataHash () const`
- `void setAlternateDataHash (const QByteArray &hash)`
- `bool hasAlternateDataSize () const`
- `qint32 alternateDataSize () const`
- `void setAlternateDataSize (const qint32 size)`
- `bool hasAlternateDataBody () const`
- `const QByteArray & alternateDataBody () const`
- `void setAlternateDataBody (const QByteArray &body)`
- `bool hasResourceAttributes () const`
- `const qevercloud::ResourceAttributes & resourceAttributes () const`
- `qevercloud::ResourceAttributes & resourceAttributes ()`
- `void setResourceAttributes (const qevercloud::ResourceAttributes &attributes)`
- `void setResourceAttributes (qevercloud::ResourceAttributes &&attributes)`
- `virtual QTextStream & print (QTextStream &strm) const` override

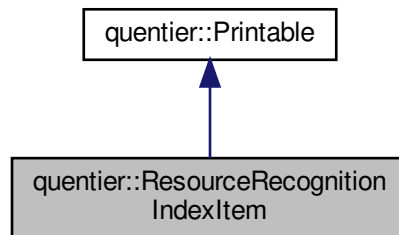
## Friends

- class `Note`

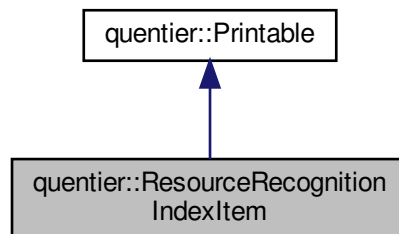
## Additional Inherited Members

### 4.58 `quentier::ResourceRecognitionIndexItem` Class Reference

Inheritance diagram for `quentier::ResourceRecognitionIndexItem`:



Collaboration diagram for `quentier::ResourceRecognitionIndexItem`:



## Classes

- struct [BarcodeItem](#)
- struct [ObjectItem](#)
- struct [ShapeItem](#)
- struct [TextItem](#)

## Public Member Functions

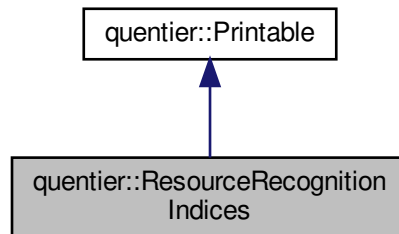
- **ResourceRecognitionIndexItem** (const [ResourceRecognitionIndexItem](#) &other)
- [ResourceRecognitionIndexItem](#) & **operator=** (const [ResourceRecognitionIndexItem](#) &other)
- bool **isValid** () const

- `int x () const`
- `void setX (const int x)`
- `int y () const`
- `void setY (const int y)`
- `int h () const`
- `void setH (const int h)`
- `int w () const`
- `void setW (const int w)`
- `int offset () const`
- `void setOffset (const int offset)`
- `int duration () const`
- `void setDuration (const int duration)`
- `QVector< int > strokeList () const`
- `int numStrokes () const`
- `bool strokeAt (const int strokeIndex, int &stroke) const`
- `bool setStrokeAt (const int strokeIndex, const int stroke)`
- `void setStrokeList (const QVector< int > &strokeList)`
- `void reserveStrokeListSpace (const int numItems)`
- `void addStroke (const int stroke)`
- `bool removeStroke (const int stroke)`
- `bool removeStrokeAt (const int strokeIndex)`
- `QVector< TextItem > textItems () const`
- `int numTextItems () const`
- `bool textItemAt (const int textItemIndex, TextItem &textItem) const`
- `bool setTextItemAt (const int textItemIndex, const TextItem &textItem)`
- `void setTextItems (const QVector< TextItem > &textItems)`
- `void reserveTextItemsSpace (const int numItems)`
- `void addTextItem (const TextItem &item)`
- `bool removeTextItem (const TextItem &item)`
- `bool removeTextItemAt (const int textItemIndex)`
- `QVector< ObjectItem > objectItems () const`
- `int numObjectItems () const`
- `bool objectItemAt (const int objectItemIndex, ObjectItem &objectItem) const`
- `bool setObjectItemAt (const int objectItemIndex, const ObjectItem &objectItem)`
- `void setObjectItems (const QVector< ObjectItem > &objectItems)`
- `void reserveObjectItemsSpace (const int numItems)`
- `void addObjectItem (const ObjectItem &item)`
- `bool removeObjectItem (const ObjectItem &item)`
- `bool removeObjectItemAt (const int objectItemIndex)`
- `QVector< ShapeItem > shapeItems () const`
- `int numShapeItems () const`
- `bool shapeItemAt (const int shapeItemIndex, ShapeItem &shapeItem) const`
- `bool setShapeItemAt (const int shapeItemIndex, const ShapeItem &shapeItem)`
- `void setShapeItems (const QVector< ShapeItem > &shapeItems)`
- `void reserveShapeItemsSpace (const int numItems)`
- `void addShapeItem (const ShapeItem &item)`
- `bool removeShapeItem (const ShapeItem &item)`
- `bool removeShapeItemAt (const int shapeItemIndex)`
- `QVector< BarcodeItem > barcodeItems () const`
- `int numBarcodeItems () const`
- `bool barcodeItemAt (const int barcodeItemIndex, BarcodeItem &barcodeItem) const`
- `bool setBarcodeItemAt (const int barcodeItemIndex, const BarcodeItem &barcodeItem)`
- `void setBarcodeItems (const QVector< BarcodeItem > &barcodeItems)`
- `void reserveBarcodeItemsSpace (const int numItems)`
- `void addBarcodeItem (const BarcodeItem &item)`
- `bool removeBarcodeItem (const BarcodeItem &item)`
- `bool removeBarcodeItemAt (const int barcodeItemIndex)`
- `virtual QTextStream & print (QTextStream &strm) const override`

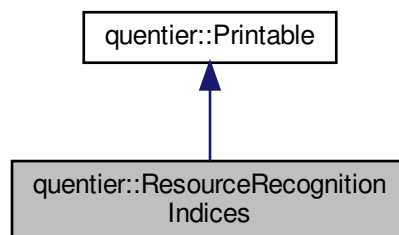
## Additional Inherited Members

### 4.59 quantier::ResourceRecognitionIndices Class Reference

Inheritance diagram for quantier::ResourceRecognitionIndices:



Collaboration diagram for quantier::ResourceRecognitionIndices:



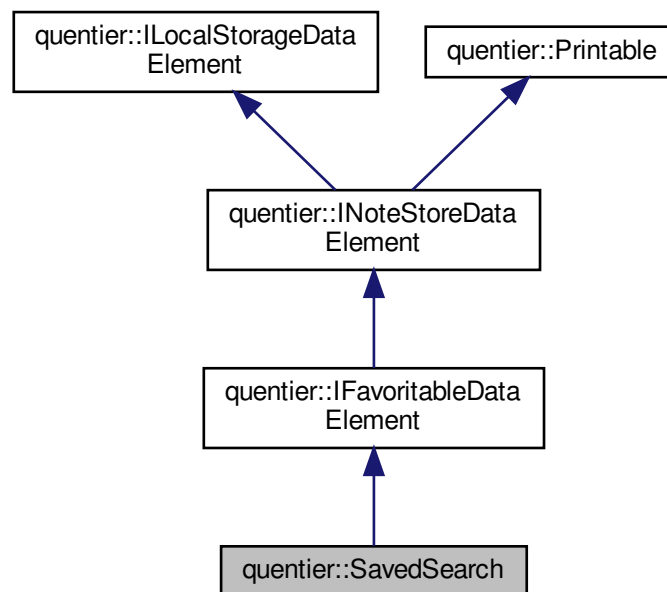
## Public Member Functions

- **ResourceRecognitionIndices** (const [ResourceRecognitionIndices](#) &other)
- **ResourceRecognitionIndices** (const QByteArray &rawRecognitionIndicesData)
- [ResourceRecognitionIndices](#) & **operator=** (const [ResourceRecognitionIndices](#) &other)
- bool **isNull** () const
- bool **isValid** () const
- QString **objectId** () const
- QString **objectType** () const
- QString **recoType** () const
- QString **engineVersion** () const
- QString **docType** () const
- QString **lang** () const
- int **objectHeight** () const
- int **objectWidth** () const
- QVector< [ResourceRecognitionIndexItem](#) > **items** () const
- bool **setData** (const QByteArray &rawRecognitionIndicesData)
- virtual QTextStream & **print** (QTextStream &strm) const override

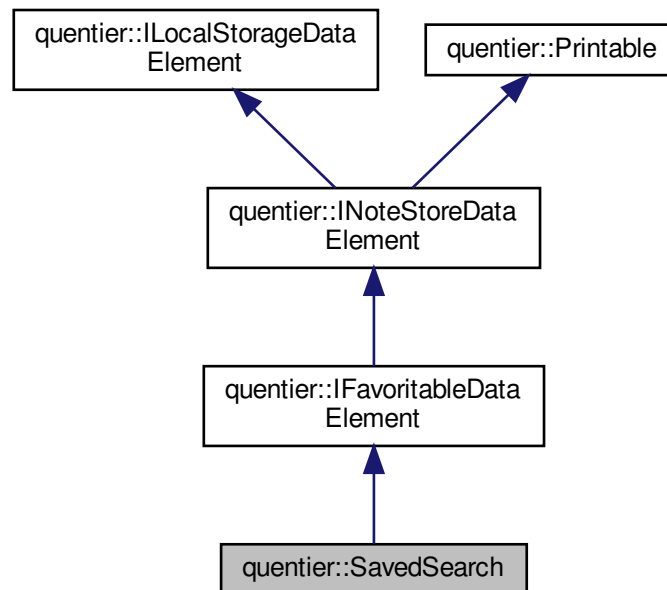
## Additional Inherited Members

### 4.60 quantier::SavedSearch Class Reference

Inheritance diagram for quantier::SavedSearch:



Collaboration diagram for `quentier::SavedSearch`:



## Public Types

- using **QueryFormat** = `qevercloud::QueryFormat`
- using **SavedSearchScope** = `qevercloud::SavedSearchScope`

## Public Member Functions

- **SavedSearch** (const [SavedSearch](#) &other)
- **SavedSearch** ([SavedSearch](#) &&other)
- [SavedSearch](#) & **operator=** (const [SavedSearch](#) &other)
- [SavedSearch](#) & **operator=** ([SavedSearch](#) &&other)
- **SavedSearch** (const `qevercloud::SavedSearch` &search)
- **SavedSearch** (`qevercloud::SavedSearch` &&search)
- const `qevercloud::SavedSearch` & **qevercloudSavedSearch** () const
- `qevercloud::SavedSearch` & **qevercloudSavedSearch** ()
- bool **operator==** (const [SavedSearch](#) &other) const
- bool **operator!=** (const [SavedSearch](#) &other) const
- virtual void **clear** () override
- virtual bool **hasGuid** () const override
- virtual const `QString` & **guid** () const override
- virtual void **setGuid** (const `QString` &guid) override
- virtual bool **hasUpdateSequenceNumber** () const override
- virtual `qint32` **updateSequenceNumber** () const override
- virtual void **setUpdateSequenceNumber** (const `qint32` usn) override
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const override



- `bool hasName () const`
- `const QString & name () const`
- `void setName (const QString &name)`
- `bool hasQuery () const`
- `const QString & query () const`
- `void setQuery (const QString &query)`
- `bool hasQueryFormat () const`
- `QueryFormat queryFormat () const`
- `void setQueryFormat (const quint8 queryFormat)`
- `bool hasIncludeAccount () const`
- `bool includeAccount () const`
- `void setIncludeAccount (const bool includeAccount)`
- `bool hasIncludePersonalLinkedNotebooks () const`
- `bool includePersonalLinkedNotebooks () const`
- `void setIncludePersonalLinkedNotebooks (const bool includePersonalLinkedNotebooks)`
- `bool hasIncludeBusinessLinkedNotebooks () const`
- `bool includeBusinessLinkedNotebooks () const`
- `void setIncludeBusinessLinkedNotebooks (const bool includeBusinessLinkedNotebooks)`
- `virtual QTextStream & print (QTextStream &strm) const` override

### Static Public Member Functions

- `static bool validateName (const QString &name, ErrorString *pErrorDescription=nullptr)`

### Additional Inherited Members

## 4.61 `quentier::ResourceRecognitionIndexItem::ShapeItem` Struct Reference

### Public Member Functions

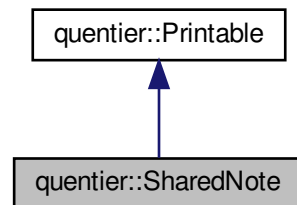
- `bool operator== (const ShapeItem &other) const`

### Public Attributes

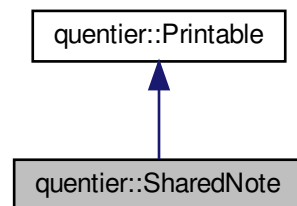
- `QString m_shapeType`
- `int m_weight = -1`

## 4.62 quantier::SharedNote Class Reference

Inheritance diagram for quantier::SharedNote:



Collaboration diagram for quantier::SharedNote:



### Public Types

- using **SharedNotePrivilegeLevel** = qevercloud::SharedNotePrivilegeLevel
- using **ContactType** = qevercloud::ContactType

### Public Member Functions

- **SharedNote** (const [SharedNote](#) &other)
- **SharedNote** ([SharedNote](#) &&other)
- **SharedNote** (const qevercloud::SharedNote &sharedNote)
- [SharedNote](#) & **operator=** (const [SharedNote](#) &other)
- [SharedNote](#) & **operator=** ([SharedNote](#) &&other)
- bool **operator==** (const [SharedNote](#) &other) const
- bool **operator!=** (const [SharedNote](#) &other) const
- const qevercloud::SharedNote & **qevercloudSharedNote** () const
- qevercloud::SharedNote & **qevercloudSharedNote** ()
- const QString & **noteGuid** () const

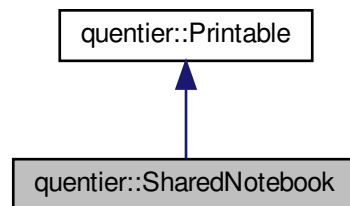
- void **setNoteGuid** (const QString &noteGuid)
- int **indexInNote** () const
- void **setIndexInNote** (const int index)
- bool **hasSharerUserId** () const
- quint32 **sharerUserId** () const
- void **setSharerUserId** (const quint32 id)
- bool **hasRecipientIdentityId** () const
- quint64 **recipientIdentityId** () const
- void **setRecipientIdentityId** (const quint64 identityId)
- bool **hasRecipientIdentityContactName** () const
- const QString & **recipientIdentityContactName** () const
- void **setRecipientIdentityContactName** (const QString &recipientIdentityContactName)
- bool **hasRecipientIdentityContactId** () const
- const QString & **recipientIdentityContactId** () const
- void **setRecipientIdentityContactId** (const QString &recipientIdentityContactId)
- bool **hasRecipientIdentityContactType** () const
- ContactType **recipientIdentityContactType** () const
- void **setRecipientIdentityContactType** (const ContactType recipientIdentityContactType)
- void **setRecipientIdentityContactType** (const quint32 recipientIdentityContactType)
- bool **hasRecipientIdentityContactPhotoUrl** () const
- const QString & **recipientIdentityContactPhotoUrl** () const
- void **setRecipientIdentityContactPhotoUrl** (const QString &recipientIdentityPhotoUrl)
- bool **hasRecipientIdentityContactPhotoLastUpdated** () const
- quint64 **recipientIdentityContactPhotoLastUpdated** () const
- void **setRecipientIdentityContactPhotoLastUpdated** (const quint64 recipientIdentityPhotoLastUpdated)
- bool **hasRecipientIdentityContactMessagingPermit** () const
- const QByteArray & **recipientIdentityContactMessagingPermit** () const
- void **setRecipientIdentityContactMessagingPermit** (const QByteArray &messagingPermit)
- bool **hasRecipientIdentityContactMessagingPermitExpires** () const
- quint64 **recipientIdentityContactMessagingPermitExpires** () const
- void **setRecipientIdentityContactMessagingPermitExpires** (const quint64 timestamp)
- bool **hasRecipientIdentityUserId** () const
- quint32 **recipientIdentityUserId** () const
- void **setRecipientIdentityUserId** (const quint32 userId)
- bool **hasRecipientIdentityDeactivated** () const
- bool **recipientIdentityDeactivated** () const
- void **setRecipientIdentityDeactivated** (const bool deactivated)
- bool **hasRecipientIdentitySameBusiness** () const
- bool **recipientIdentitySameBusiness** () const
- void **setRecipientIdentitySameBusiness** (const bool sameBusiness)
- bool **hasRecipientIdentityBlocked** () const
- bool **recipientIdentityBlocked** () const
- void **setRecipientIdentityBlocked** (const bool blocked)
- bool **hasRecipientIdentityUserConnected** () const
- bool **recipientIdentityUserConnected** () const
- void **setRecipientIdentityUserConnected** (const bool userConnected)
- bool **hasRecipientIdentityEventId** () const
- quint64 **recipientIdentityEventId** () const
- void **setRecipientIdentityEventId** (const quint64 eventId)
- bool **hasRecipientIdentity** () const
- const qevercloud::Identity & **recipientIdentity** () const
- void **setRecipientIdentity** (qevercloud::Identity &&identity)
- bool **hasPrivilegeLevel** () const
- SharedNotePrivilegeLevel **privilegeLevel** () const
- void **setPrivilegeLevel** (const SharedNotePrivilegeLevel level)

- void **setPrivilegeLevel** (const qint8 level)
- bool **hasCreationTimestamp** () const
- qint64 **creationTimestamp** () const
- void **setCreationTimestamp** (const qint64 timestamp)
- bool **hasModificationTimestamp** () const
- qint64 **modificationTimestamp** () const
- void **setModificationTimestamp** (const qint64 timestamp)
- bool **hasAssignmentTimestamp** () const
- qint64 **assignmentTimestamp** () const
- void **setAssignmentTimestamp** (const qint64 timestamp)
- virtual QTextStream & **print** (QTextStream &strm) const override

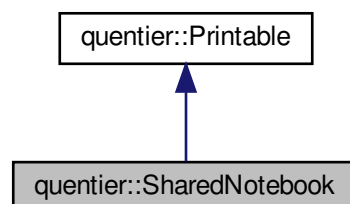
## Additional Inherited Members

## 4.63 quantier::SharedNotebook Class Reference

Inheritance diagram for quantier::SharedNotebook:



Collaboration diagram for quantier::SharedNotebook:



## Public Types

- using **SharedNotebookPrivilegeLevel** = qevercloud::SharedNotebookPrivilegeLevel

## Public Member Functions

- **SharedNotebook** (const [SharedNotebook](#) &other)
- **SharedNotebook** ([SharedNotebook](#) &&other)
- **SharedNotebook** (const qevercloud::SharedNotebook &qecSharedNotebook)
- [SharedNotebook](#) & **operator=** (const [SharedNotebook](#) &other)
- [SharedNotebook](#) & **operator=** ([SharedNotebook](#) &&other)
- bool **operator==** (const [SharedNotebook](#) &other) const
- bool **operator!=** (const [SharedNotebook](#) &other) const
- const qevercloud::SharedNotebook & **qevercloudSharedNotebook** () const
- qevercloud::SharedNotebook & **qevercloudSharedNotebook** ()
- int **indexInNotebook** () const
- void **setIndexInNotebook** (const int index)
- bool **hasId** () const
- quint64 **id** () const
- void **setId** (const quint64 id)
- bool **hasUserId** () const
- quint32 **userId** () const
- void **setUserId** (const quint32 userId)
- bool **hasNotebookGuid** () const
- const QString & **notebookGuid** () const
- void **setNotebookGuid** (const QString &notebookGuid)
- bool **hasEmail** () const
- const QString & **email** () const
- void **setEmail** (const QString &email)
- bool **hasCreationTimestamp** () const
- quint64 **creationTimestamp** () const
- void **setCreationTimestamp** (const quint64 timestamp)
- bool **hasModificationTimestamp** () const
- quint64 **modificationTimestamp** () const
- void **setModificationTimestamp** (const quint64 timestamp)
- bool **hasUsername** () const
- const QString & **username** () const
- void **setUsername** (const QString &username)
- bool **hasPrivilegeLevel** () const
- SharedNotebookPrivilegeLevel **privilegeLevel** () const
- void **setPrivilegeLevel** (const SharedNotebookPrivilegeLevel privilegeLevel)
- void **setPrivilegeLevel** (const quint8 privilegeLevel)
- bool **hasReminderNotifyEmail** () const
- bool **reminderNotifyEmail** () const
- void **setReminderNotifyEmail** (const bool notifyEmail)
- bool **hasReminderNotifyApp** () const
- bool **reminderNotifyApp** () const
- void **setReminderNotifyApp** (const bool notifyApp)
- bool **hasRecipientUsername** () const
- const QString & **recipientUsername** () const
- void **setRecipientUsername** (const QString &recipientUsername)
- bool **hasRecipientUserId** () const
- quint32 **recipientUserId** () const
- void **setRecipientUserId** (const quint32 userId)
- bool **hasRecipientIdentityId** () const
- quint64 **recipientIdentityId** () const
- void **setRecipientIdentityId** (const quint64 recipientIdentityId)
- bool **hasGlobalId** () const
- const QString & **globalId** () const

- void **setGlobalId** (const QString &globalId)
- bool **hasSharerUserId** () const
- qint32 **sharerUserId** () const
- void **setSharerUserId** (qint32 sharerUserId)
- bool **hasAssignmentTimestamp** () const
- qint64 **assignmentTimestamp** () const
- void **setAssignmentTimestamp** (const qint64 timestamp)
- virtual QTextStream & **print** (QTextStream &strm) const override

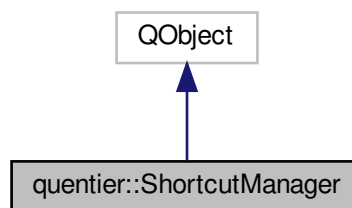
## Friends

- class **Notebook**

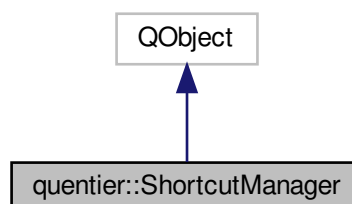
## Additional Inherited Members

## 4.64 quantier::ShortcutManager Class Reference

Inheritance diagram for quantier::ShortcutManager:



Collaboration diagram for quantier::ShortcutManager:



## Public Types

- enum **QuentierShortcutKey** {  
**NewNote** = 5000 , **NewTag** , **NewNotebook** , **NewSavedSearch** ,  
**AddAttachment** , **SaveAttachment** , **OpenAttachment** , **CopyAttachment** ,  
**CutAttachment** , **RemoveAttachment** , **RenameAttachment** , **AddAccount** ,  
**ExitAccount** , **SwitchAccount** , **AccountInfo** , **NoteSearch** ,  
**NewNoteSearch** , **ShowNotes** , **ShowNotebooks** , **ShowTags** ,  
**ShowSavedSearches** , **ShowDeletedNotes** , **ShowStatusBar** , **ShowToolBar** ,  
**PasteUnformatted** , **Font** , **UpperIndex** , **LowerIndex** ,  
**AlignLeft** , **AlignCenter** , **AlignRight** , **AlignFull** ,  
**IncreaseIndentation** , **DecreaseIndentation** , **IncreaseFontSize** , **DecreaseFontSize** ,  
**InsertNumberedList** , **InsertBulletedList** , **Strikethrough** , **Highlight** ,  
**InsertTable** , **InsertRow** , **InsertColumn** , **RemoveRow** ,  
**RemoveColumn** , **InsertHorizontalLine** , **InsertToDoTag** , **EditHyperlink** ,  
**CopyHyperlink** , **RemoveHyperlink** , **Encrypt** , **Decrypt** ,  
**DecryptPermanently** , **BackupLocalStorage** , **RestoreLocalStorage** , **UpgradeLocalStorage** ,  
**LocalStorageStatus** , **SpellCheck** , **SpellCheckIgnoreWord** , **SpellCheckAddWordToUserDictionary** ,  
**SaveImage** , **AnnotateImage** , **ImageRotateClockwise** , **ImageRotateCounterClockwise** ,  
**Synchronize** , **FullSync** , **ImportFolders** , **Preferences** ,  
**ReleaseNotes** , **ViewLogs** , **About** , **UnknownKey** = 100000 }

## Public Slots

- void **setUserShortcut** (int key, QKeySequence [shortcut](#), const [Account](#) &account, QString context={})
- void **setNonStandardUserShortcut** (QString nonStandardKey, QKeySequence [shortcut](#), const [Account](#) &account, QString context={})
- void **setDefaultShortcut** (int key, QKeySequence [shortcut](#), const [Account](#) &account, QString context={})
- void **setNonStandardDefaultShortcut** (QString nonStandardKey, QKeySequence [shortcut](#), const [Account](#) &account, QString context={})

## Signals

- void **shortcutChanged** (int key, QKeySequence [shortcut](#), const [Account](#) &account, QString context)
- void **nonStandardShortcutChanged** (QString nonStandardKey, QKeySequence [shortcut](#), const [Account](#) &account, QString context)

## Public Member Functions

- ShortcutManager** (QObject \*parent=nullptr)
- QKeySequence [shortcut](#) (const int key, const [Account](#) &account, const QString &context={}) const
- QKeySequence [shortcut](#) (const QString &nonStandardKey, const [Account](#) &account, const QString &context={}) const
- QKeySequence [defaultShortcut](#) (const int key, const [Account](#) &account, const QString &context={}) const
- QKeySequence [defaultShortcut](#) (const QString &nonStandardKey, const [Account](#) &account, const QString &context={}) const
- QKeySequence [userShortcut](#) (const int key, const [Account](#) &account, const QString &context={}) const
- QKeySequence [userShortcut](#) (const QString &nonStandardKey, const [Account](#) &account, const QString &context={}) const

### 4.64.1 Member Function Documentation

#### 4.64.1.1 defaultShortcut() [1/2]

```
QKeySequence quentier::ShortcutManager::defaultShortcut (
    const int key,
    const Account & account,
    const QString & context = {} ) const
```

##### Returns

Default shortcut for the standard key if present, otherwise empty key sequence

#### 4.64.1.2 defaultShortcut() [2/2]

```
QKeySequence quentier::ShortcutManager::defaultShortcut (
    const QString & nonStandardKey,
    const Account & account,
    const QString & context = {} ) const
```

##### Returns

Default shortcut for the non-standard key if present, otherwise empty key sequence

#### 4.64.1.3 shortcut() [1/2]

```
QKeySequence quentier::ShortcutManager::shortcut (
    const int key,
    const Account & account,
    const QString & context = {} ) const
```

##### Returns

Active shortcut for the standard key - either the user defined shortcut (if present) or the default one (if present as well)

#### 4.64.1.4 shortcut() [2/2]

```
QKeySequence quentier::ShortcutManager::shortcut (
    const QString & nonStandardKey,
    const Account & account,
    const QString & context = {} ) const
```

##### Returns

Active shortcut for the non-standard key - either the user defined shortcut (if present) or the default one (if present as well)



## 4.64.1.5 userShortcut() [1/2]

```
QKeySequence quantier::ShortcutManager::userShortcut (
    const int key,
    const Account & account,
    const QString & context = {} ) const
```

## Returns

User defined shortcut for the standard key if present, otherwise empty key sequence

## 4.64.1.6 userShortcut() [2/2]

```
QKeySequence quantier::ShortcutManager::userShortcut (
    const QString & nonStandardKey,
    const Account & account,
    const QString & context = {} ) const
```

## Returns

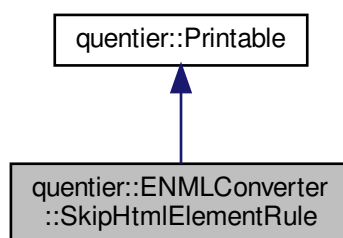
User defined shortcut for the non-standard key if present, otherwise empty key sequence

## 4.65 quantier::ENMLConverter::SkipHtmlElementRule Class Reference

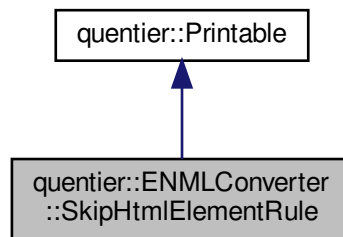
The [SkipHtmlElementRule](#) class describes the set of rules for HTML -> ENML conversion about the HTML elements that should not be actually converted to ENML due to their nature of being "helper" elements for the display or functioning of something within the note editor's page. The HTML -> ENML conversion would ignore tags and attributes forbidden by ENML even without these rules conditionally preserving or skipping the contents and nested elements of skipped elements.

```
#include <ENMLConverter.h>
```

Inheritance diagram for quantier::ENMLConverter::SkipHtmlElementRule:



Collaboration diagram for `quentier::ENMLConverter::SkipHtmlElementRule`:



## Public Types

- enum class **ComparisonRule** { **Equals** = 0 , **StartsWith** , **EndsWith** , **Contains** }

## Public Member Functions

- virtual `QTextStream & print (QTextStream &strm)` const override

## Public Attributes

- `QString m_elementNameToSkip`
- `ComparisonRule m_elementNameComparisonRule = ComparisonRule::Equals`
- `Qt::CaseSensitivity m_elementNameCaseSensitivity = Qt::CaseSensitive`
- `QString m_attributeNameToSkip`
- `ComparisonRule m_attributeNameComparisonRule = ComparisonRule::Equals`
- `Qt::CaseSensitivity m_attributeNameCaseSensitivity = Qt::CaseSensitive`
- `QString m_attributeValueToSkip`
- `ComparisonRule m_attributeValueComparisonRule = ComparisonRule::Equals`
- `Qt::CaseSensitivity m_attributeValueCaseSensitivity = Qt::CaseSensitive`
- `bool m_includeElementContents = false`

## Friends

- `QUENTIER_EXPORT QTextStream & operator<< (QTextStream &strm, const ComparisonRule rule)`

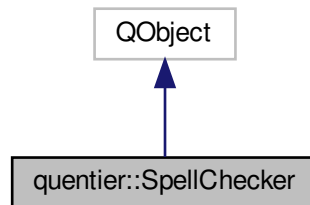
## Additional Inherited Members

### 4.65.1 Detailed Description

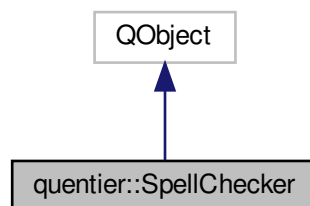
The [SkipHtmlElementRule](#) class describes the set of rules for HTML -> ENML conversion about the HTML elements that should not be actually converted to ENML due to their nature of being "helper" elements for the display or functioning of something within the note editor's page. The HTML -> ENML conversion would ignore tags and attributes forbidden by ENML even without these rules conditionally preserving or skipping the contents and nested elements of skipped elements.

## 4.66 quantier::SpellChecker Class Reference

Inheritance diagram for quantier::SpellChecker:



Collaboration diagram for quantier::SpellChecker:



### Signals

- void **ready** ()

### Public Member Functions

- **SpellChecker** ([FileIOProcessorAsync](#) \*pFileIOProcessorAsync, const [Account](#) &account, QObject \*parent=nullptr, const QString &userDictionaryPath={})
- QVector< std::pair< QString, bool > > **listAvailableDictionaries** () const
- void **setAccount** (const [Account](#) &account)
- void **enableDictionary** (const QString &language)
- void **disableDictionary** (const QString &language)
- bool **checkSpell** (const QString &word) const
- QStringList **spellCorrectionSuggestions** (const QString &misSpelledWord) const
- void **addToUserWordlist** (const QString &word)
- void **removeFromUserWordList** (const QString &word)
- void **ignoreWord** (const QString &word)
- void **removeWord** (const QString &word)
- bool **isReady** () const

## 4.67 `quentier::StringUtils::StringFilterPredicate` Struct Reference

### Public Member Functions

- **StringFilterPredicate** (QSet< QString > &filteredStrings)
- **bool operator()** (const QString &str) const

### Public Attributes

- QSet< QString > & **m\_filteredStrings**

## 4.68 `quentier::StringUtils` Class Reference

### Classes

- struct [StringFilterPredicate](#)

### Public Member Functions

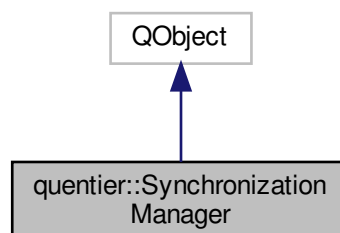
- void **removePunctuation** (QString &str, const QVector< QChar > &charactersToPreserve={}) const
- void **removeDiacritics** (QString &str) const
- void **removeNewlines** (QString &str) const

## 4.69 `quentier::SynchronizationManager` Class Reference

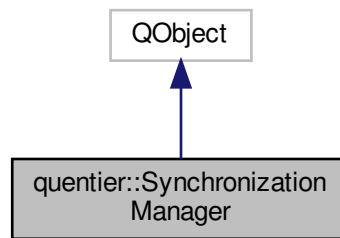
The [SynchronizationManager](#) class encapsulates methods and signals & slots required to perform the full or partial synchronization of data with remote Evernote servers. The class also deals with authentication with Evernote service through OAuth.

```
#include <SynchronizationManager.h>
```

Inheritance diagram for `quentier::SynchronizationManager`:



Collaboration diagram for quantier::SynchronizationManager:



## Public Slots

- void [setAccount](#) ([Account](#) account)
- void [authenticate](#) ()
- void [authenticateCurrentAccount](#) ()
- void [synchronize](#) ()
- void [stop](#) ()
- void [revokeAuthentication](#) (const [qevercloud::UserID](#) userId)
- void [setDownloadNoteThumbnails](#) (bool flag)
- void [setDownloadInkNoteImages](#) (bool flag)
- void [setInkNoteImagesStoragePath](#) ([QString](#) path)

## Signals

- void [started](#) ()
- void [stopped](#) ()
- void [failed](#) ([ErrorString](#) errorDescription)
- void [finished](#) ([Account](#) account, bool somethingDownloaded, bool somethingSent)
- void [authenticationRevoked](#) (bool success, [ErrorString](#) errorDescription, [qevercloud::UserID](#) userId)
- void [authenticationFinished](#) (bool success, [ErrorString](#) errorDescription, [Account](#) account)
- void [remoteToLocalSyncStopped](#) ()
- void [sendLocalChangesStopped](#) ()
- void [willRepeatRemoteToLocalSyncAfterSendingChanges](#) ()
- void [detectedConflictDuringLocalChangesSending](#) ()
- void [rateLimitExceeded](#) (qint32 secondsToWait)
- void [remoteToLocalSyncDone](#) (bool somethingDownloaded)
- void [syncChunksDownloadProgress](#) (qint32 highestDownloadedUsn, qint32 highestServerUsn, qint32 last← PreviousUsn)
- void [syncChunksDownloaded](#) ()
- void [linkedNotebookSyncChunksDownloadProgress](#) (qint32 highestDownloadedUsn, qint32 highestServer← Usn, qint32 lastPreviousUsn, [LinkedNotebook](#) linkedNotebook)
- void [linkedNotebooksSyncChunksDownloaded](#) ()
- void [notesDownloadProgress](#) (quint32 notesDownloaded, quint32 totalNotesToDownload)
- void [linkedNotebooksNotesDownloadProgress](#) (quint32 notesDownloaded, quint32 totalNotesToDownload)
- void [resourcesDownloadProgress](#) (quint32 resourcesDownloaded, quint32 totalResourcesToDownload)

- void [linkedNotebooksResourcesDownloadProgress](#) (quint32 resourcesDownloaded, quint32 total↔ ResourcesToDownload)
- void [preparedDirtyObjectsForSending](#) ()
- void [preparedLinkedNotebooksDirtyObjectsForSending](#) ()
- void [setAccountDone](#) (Account account)
- void [setDownloadNoteThumbnailsDone](#) (bool flag)
- void [setDownloadInkNoteImagesDone](#) (bool flag)
- void [setInkNoteImagesStoragePathDone](#) (QString path)

## Public Member Functions

- [SynchronizationManager](#) (QString host, [LocalStorageManagerAsync](#) &localStorageManagerAsync, [IAuthenticationManager](#) &authenticationManager, QObject \*parent=nullptr, INoteStorePtr pNoteStore={}, IUserStorePtr pUserStore={}, IKeychainServicePtr pKeychainService={}, ISyncStateStoragePtr pSync↔ StateStorage={})
- bool [active](#) () const
- bool [downloadNoteThumbnailsOption](#) () const

### 4.69.1 Detailed Description

The [SynchronizationManager](#) class encapsulates methods and signals & slots required to perform the full or partial synchronization of data with remote Evernote servers. The class also deals with authentication with Evernote service through OAuth.

### 4.69.2 Constructor & Destructor Documentation

#### 4.69.2.1 SynchronizationManager()

```
quentier::SynchronizationManager::SynchronizationManager (
    QString host,
    LocalStorageManagerAsync & localStorageManagerAsync,
    IAuthenticationManager & authenticationManager,
    QObject * parent = nullptr,
    INoteStorePtr pNoteStore = {},
    IUserStorePtr pUserStore = {},
    IKeychainServicePtr pKeychainService = {},
    ISyncStateStoragePtr pSyncStateStorage = {} )
```

#### Parameters

<i>host</i>	The host to use for the connection with the Evernote service - typically <a href="#">www.evernote.com</a> but could also be <a href="#">sandbox.evernote.com</a> or some other one
<i>localStorageManagerAsync</i>	Local storage manager
<i>authenticationManager</i>	Reference to an object implementing <a href="#">IAuthenticationManager</a> interface; <a href="#">SynchronizationManager</a> does not store this reference, it only connects to the object via signals and slots during construction
<i>pNoteStore</i>	Pointer to an object implementing <a href="#">INoteStore</a> interface; if nullptr, SynchronizatrionManager would create and use its own instance; otherwise <a href="#">SynchronizationManager</a> would set itself as the parent of the passed in object

## Parameters

<i>pUserStore</i>	Pointer to an object implementing <a href="#">IUserStore</a> interface; if nullptr, <a href="#">SynchronizationManager</a> would create and use its own instance
<i>pKeychainService</i>	Pointer to an object implementing <a href="#">IKeychainService</a> interface; if nullptr, <a href="#">SynchronizationManager</a> would create and use its own default instance; otherwise <a href="#">SynchronizationManager</a> would set itself as the parent of the passed in object
<i>pSyncStateStorage</i>	Pointer to an object implementing <a href="#">ISyncStateStorage</a> interface; if nullptr, <a href="#">SynchronizationManager</a> would create and use its own instance; otherwise <a href="#">SynchronizationManager</a> would set itself as the parent of the passed in object

### 4.69.3 Member Function Documentation

#### 4.69.3.1 active()

```
bool quantier::SynchronizationManager::active ( ) const
```

## Returns

True if the synchronization is being performed at the moment, false otherwise

#### 4.69.3.2 authenticate

```
void quantier::SynchronizationManager::authenticate ( ) [slot]
```

Use this slot to authenticate the new user to do the synchronization with the Evernote service via the client app. The invocation of this slot would respond asynchronously with authenticationFinished signal but won't start the synchronization.

**Note** that this slot would always proceed to the actual OAuth.

#### 4.69.3.3 authenticateCurrentAccount

```
void quantier::SynchronizationManager::authenticateCurrentAccount ( ) [slot]
```

Use this slot to authenticate the current account to do the synchronization with the Evernote service via the client app. The invocation of this slot would respond asynchronously with authenticationFinished signal but won't start the synchronization

If no account was set to [SynchronizationManager](#) prior to this slot invocation, it would proceed to OAuth. Otherwise [SynchronizationManager](#) would first check whether the persistent authentication data is in place and actual. If yes, no OAuth would be performed.

#### 4.69.3.4 authenticationFinished

```
void quantier::SynchronizationManager::authenticationFinished (
    bool success,
    ErrorString errorDescription,
    Account account ) [signal]
```

This signal is emitted in response to the explicit attempt to authenticate the new user of the client app to the Evernote service. NOTE: this signal is not emitted if the authentication was requested automatically during sync attempt, it is only emitted in response to the explicit invocation of authenticate slot.

## Parameters

<i>success</i>	True if the authentication was successful, false otherwise
<i>errorDescription</i>	The textual explanation of the failure to authenticate the new user
<i>account</i>	The account of the authenticated user

**4.69.3.5 authenticationRevoked**

```
void quentier::SynchronizationManager::authenticationRevoked (
    bool success,
    ErrorString errorDescription,
    qevercloud::UserID userId ) [signal]
```

This signal is emitted in response to the attempt to revoke the authentication for a given user ID

## Parameters

<i>success</i>	True if the authentication was revoked successfully, false otherwise
<i>errorDescription</i>	The textual explanation of the failure to revoke the authentication
<i>userId</i>	The ID of the user for which the revoke of the authentication was requested

**4.69.3.6 detectedConflictDuringLocalChangesSending**

```
void quentier::SynchronizationManager::detectedConflictDuringLocalChangesSending ( ) [signal]
```

This signal is emitted if during the "send local changes" synchronization step it was found out that new changes from the Evernote service are available AND some of them conflict with the local changes being sent.

Such situation can rarely happen in case of changes introduced concurrently with the running synchronization - perhaps via another client. The algorithm will handle it by repeating the "remote to local" incremental synchronization step, the signal is just for the sake of diagnostic.

**4.69.3.7 downloadNoteThumbnailsOption()**

```
bool quentier::SynchronizationManager::downloadNoteThumbnailsOption ( ) const
```

## Returns

True or false depending on the option to download the thumbnails for notes containing resources during sync; by default no thumbnails are downloaded



#### 4.69.3.8 failed

```
void quantier::SynchronizationManager::failed (
    ErrorString errorDescription ) [signal]
```

This signal is emitted when the synchronization fails; at this moment there is no error code explaining the reason of the failure programmatically so the only explanation available is the textual one for the end user

#### 4.69.3.9 finished

```
void quantier::SynchronizationManager::finished (
    Account account,
    bool somethingDownloaded,
    bool somethingSent ) [signal]
```

This signal is emitted when the synchronization is finished

##### Parameters

<i>account</i>	Represents the latest version of <a href="#">Account</a> structure filled during the synchronization procedure
<i>somethingDownloaded</i>	Boolean parameter telling the receiver whether any data items were actually downloaded during remote to local synchronization step; if there was nothing to sync up from the remote storage, this boolean would be false, otherwise it would be true
<i>somethingSent</i>	Boolean parameter telling the receiver whether any data items were actually sent during the send local changes synchronization step; if there was nothing to send to the remote storage, this boolean would be false, otherwise it would be true

#### 4.69.3.10 linkedNotebooksNotesDownloadProgress

```
void quantier::SynchronizationManager::linkedNotebooksNotesDownloadProgress (
    quint32 notesDownloaded,
    quint32 totalNotesToDownload ) [signal]
```

This signal is emitted on each successful download of full note data from linked notebooks.

##### Parameters

<i>notesDownloaded</i>	The number of notes downloaded by the moment
<i>totalNotesToDownload</i>	The total number of notes that need to be downloaded

#### 4.69.3.11 linkedNotebooksResourcesDownloadProgress

```
void quantier::SynchronizationManager::linkedNotebooksResourcesDownloadProgress (
    quint32 resourcesDownloaded,
    quint32 totalResourcesToDownload ) [signal]
```

This signal is emitted on each successful download of full resource data from linked notebooks during the incremental sync (as individual resources are downloaded along with their notes during full sync).

#### Parameters

<i>resourcesDownloaded</i>	The number of resources downloaded by the moment
<i>totalResourcesToDownload</i>	The total number of resources that need to be downloaded

#### 4.69.3.12 linkedNotebooksSyncChunksDownloaded

```
void quantier::SynchronizationManager::linkedNotebooksSyncChunksDownloaded ( ) [signal]
```

This signal is emitted when the sync chunks for the stuff from linked notebooks are downloaded during "remote to local" synchronization step

#### 4.69.3.13 linkedNotebookSyncChunksDownloadProgress

```
void quantier::SynchronizationManager::linkedNotebookSyncChunksDownloadProgress (
    quint32 highestDownloadedUsn,
    quint32 highestServerUsn,
    quint32 lastPreviousUsn,
    LinkedNotebook linkedNotebook ) [signal]
```

This signal is emitted during linked notebooks sync chunks downloading and denotes the progress of that step, individually for each linked notebook. The percentage of completeness can be computed roughly as  $(\text{highestDownloadedUsn} - \text{lastPreviousUsn}) / (\text{highestServerUsn} - \text{lastPreviousUsn}) * 100\%$ . The sync chunks for each linked notebook are downloaded sequentially so the signals for one linked notebook should not intermix with signals for other linked notebooks, however, it is within hands of Qt's slot schedulers

#### Parameters

<i>highestDownloadedUsn</i>	The highest update sequence number within data items from linked notebook sync chunks downloaded so far
<i>highestServerUsn</i>	The current highest update sequence number within the linked notebook
<i>lastPreviousUsn</i>	The last update sequence number from previous sync of the given linked notebook; if current sync is the first one, this value is zero
<i>linkedNotebook</i>	The linked notebook which sync chunks download progress is reported

#### 4.69.3.14 notesDownloadProgress

```
void quantier::SynchronizationManager::notesDownloadProgress (
    quint32 notesDownloaded,
    quint32 totalNotesToDownload ) [signal]
```

This signal is emitted on each successful download of full note data from use 's own account.

## Parameters

<i>notesDownloaded</i>	The number of notes downloaded by the moment
<i>totalNotesToDownload</i>	The total number of notes that need to be downloaded

**4.69.3.15 preparedDirtyObjectsForSending**

```
void quantier::SynchronizationManager::preparedDirtyObjectsForSending ( ) [signal]
```

This signal is emitted during "send local changes" synchronization step when all the relevant data elements from user's own account were prepared for sending to the Evernote service

**4.69.3.16 preparedLinkedNotebooksDirtyObjectsForSending**

```
void quantier::SynchronizationManager::preparedLinkedNotebooksDirtyObjectsForSending ( ) [signal]
```

This signal is emitted during "send local changes" synchronization step when all the relevant data elements from linked notebooks were prepared for sending to the Evernote service

**4.69.3.17 rateLimitExceeded**

```
void quantier::SynchronizationManager::rateLimitExceeded (
    qint32 secondsToWait ) [signal]
```

This signal is emitted when the Evernote API rate limit is breached during the synchronization; the algorithm will handle it by auto-pausing itself until the time necessary to wait passes and then automatically continuing the synchronization.

## Parameters

<i>secondsToWait</i>	The amount of time (in seconds) necessary to wait before the synchronization will continue
----------------------	--

**4.69.3.18 remoteToLocalSyncDone**

```
void quantier::SynchronizationManager::remoteToLocalSyncDone (
    bool somethingDownloaded ) [signal]
```

This signal is emitted when the "remote to local" synchronization step is finished; once that step is done, the algorithm switches to sending the local changes back to the Evernote service.

## Parameters

<i>somethingDownloaded</i>	Boolean parameter telling the receiver whether any data items were actually downloaded during remote to local synchronization step; if there was nothing to sync up from the remote storage, this boolean would be false, otherwise it would be true
----------------------------	--

**4.69.3.19 remoteToLocalSyncStopped**

```
void quantier::SynchronizationManager::remoteToLocalSyncStopped ( ) [signal]
```

This signal is emitted when the "remote to local" synchronization step is stopped

**4.69.3.20 resourcesDownloadProgress**

```
void quantier::SynchronizationManager::resourcesDownloadProgress (
    quint32 resourcesDownloaded,
    quint32 totalResourcesToDownload ) [signal]
```

This signal is emitted on each successful download of full resource data from user's own account during the incremental sync (as individual resources are downloaded along with their notes during full sync).

## Parameters

<i>resourcesDownloaded</i>	The number of resources downloaded by the moment
<i>totalResourcesToDownload</i>	The total number of resources that need to be downloaded

**4.69.3.21 revokeAuthentication**

```
void quantier::SynchronizationManager::revokeAuthentication (
    const qevercloud::UserID userId ) [slot]
```

Use this slot to remove any previously cached authentication tokens (and shard ids) for a given user ID. After the call of this method the next attempt to synchronize the data for this user ID would cause the launch of OAuth to get the new authentication token

**4.69.3.22 sendLocalChangesStopped**

```
void quantier::SynchronizationManager::sendLocalChangesStopped ( ) [signal]
```

This signal is emitted when the "send local changes" synchronization step is stopped

#### 4.69.3.23 setAccount

```
void quantier::SynchronizationManager::setAccount (
    Account account ) [slot]
```

Use this slot to set the current account for the synchronization manager. If the slot is called during the synchronization running, it would stop, any internal caches belonging to previously selected account (if any) would be purged (but persistent settings like the authentication token saved in the system keychain would remain). Setting the current account won't automatically start the synchronization for it, use synchronize slot for this.

The attempt to set the current account of "Local" type would just clean up the synchronization manager as if it was just created

After the method finishes its job, setAccountDone signal is emitted

#### 4.69.3.24 setAccountDone

```
void quantier::SynchronizationManager::setAccountDone (
    Account account ) [signal]
```

This signal is emitted in response to invoking the setAccount slot after all the activities involved in switching the account inside [SynchronizationManager](#) are finished

#### 4.69.3.25 setDownloadInkNoteImages

```
void quantier::SynchronizationManager::setDownloadInkNoteImages (
    bool flag ) [slot]
```

Use this slot to switch the option whether the synchronization of notes would download the plain images corresponding to ink notes or not. By default the downloading of ink note images is disabled.

After the method finishes its job, setDownloadInkNoteImagesDone signal is emitted

#### 4.69.3.26 setDownloadInkNoteImagesDone

```
void quantier::SynchronizationManager::setDownloadInkNoteImagesDone (
    bool flag ) [signal]
```

This signal is emitted in response to invoking the setDownloadInkNoteImages slot after the setting is accepted

#### 4.69.3.27 setDownloadNoteThumbnails

```
void quantier::SynchronizationManager::setDownloadNoteThumbnails (
    bool flag ) [slot]
```

Use this slot to switch the option whether the synchronization of notes would download the note thumbnails or not. By default the downloading of thumbnails for notes containing resources is disabled.

NOTE: even if thumbnails downloading is enabled, the thumbnails would be downloaded during sync only for notes containing resources

After the method finishes its job, setDownloadNoteThumbnailsDone signal is emitted

#### 4.69.3.28 setDownloadNoteThumbnailsDone

```
void quantier::SynchronizationManager::setDownloadNoteThumbnailsDone (
    bool flag ) [signal]
```

This signal is emitted in response to invoking the setDownloadNoteThumbnails slot after the setting is accepted

#### 4.69.3.29 setInkNoteImagesStoragePath

```
void quantier::SynchronizationManager::setInkNoteImagesStoragePath (
    QString path ) [slot]
```

Use this slot to specify the path to folder at which the downloaded ink note images should be stored. Each ink note image would be stored in a separate PNG file which name would be the same as the guid of the corresponding resource and the file extension would be PNG

The default storage path would be the folder "inkNoteImages" within the folder returned by applicationPersistentStoragePath function found in quantier/StandardPaths.h header

WARNING: if the passed in path cannot be used (either it doesn't exist and cannot be created or exists but is not writable), the default path is silently restored. So make sure you're setting a valid path

After the method finishes its job, setInkNoteImagesStoragePathDone signal is emitted

#### 4.69.3.30 setInkNoteImagesStoragePathDone

```
void quantier::SynchronizationManager::setInkNoteImagesStoragePathDone (
    QString path ) [signal]
```

This signal is emitted in response to invoking the setInkNoteImagesStoragePath slot after the setting is accepted

#### 4.69.3.31 started

```
void quantier::SynchronizationManager::started ( ) [signal]
```

This signal is emitted when the synchronization is started (authentication is not considered a part of synchronization so this signal is only emitted when the authentication is completed)

#### 4.69.3.32 stop

```
void quantier::SynchronizationManager::stop ( ) [slot]
```

Use this slot to stop the running synchronization; if no synchronization is running by the moment of this slot call, it has no effect

#### 4.69.3.33 stopped

```
void quantier::SynchronizationManager::stopped ( ) [signal]
```

This signal is emitted in response to invoking the stop slot, whether it was invoked manually or from within the [SynchronizationManager](#) itself (due to sync failure, for example)

**4.69.3.34 syncChunksDownloaded**

```
void quantier::SynchronizationManager::syncChunksDownloaded ( ) [signal]
```

This signal is emitted when the sync chunks for the stuff from user's own account are downloaded during "remote to local" synchronization step

**4.69.3.35 syncChunksDownloadProgress**

```
void quantier::SynchronizationManager::syncChunksDownloadProgress (
    quint32 highestDownloadedUsn,
    quint32 highestServerUsn,
    quint32 lastPreviousUsn ) [signal]
```

This signal is emitted during user own account's sync chunks downloading and denotes the progress of that step. The percentage of completeness can be computed roughly as  $(\text{highestDownloadedUsn} - \text{lastPreviousUsn}) / (\text{highestServerUsn} - \text{lastPreviousUsn}) * 100\%$

**Parameters**

<i>highestDownloadedUsn</i>	The highest update sequence number within data items from sync chunks downloaded so far
<i>highestServerUsn</i>	The current highest update sequence number within the account
<i>lastPreviousUsn</i>	The last update sequence number from previous sync; if current sync is the first one, this value is zero

**4.69.3.36 synchronize**

```
void quantier::SynchronizationManager::synchronize ( ) [slot]
```

Use this slot to launch the synchronization of data

**4.69.3.37 willRepeatRemoteToLocalSyncAfterSendingChanges**

```
void quantier::SynchronizationManager::willRepeatRemoteToLocalSyncAfterSendingChanges ( )
[signal]
```

This signal is emitted if during the "send local changes" synchronization step it was found out that new changes from the Evernote service are available yet no conflict between remote and local changes was found yet.

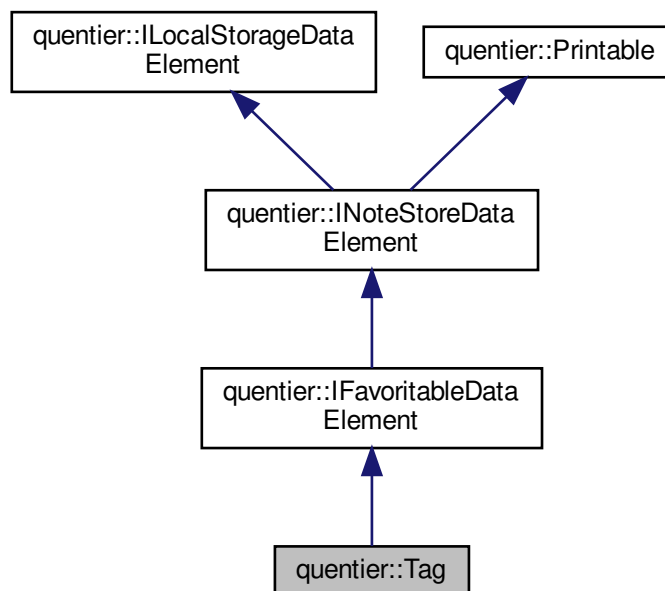
Such situation can rarely happen in case of changes introduced concurrently with the running synchronization - perhaps via another client. The algorithm will handle it, the signal is just for the sake of diagnostic.

**4.70 quantier::SysInfo Class Reference****Public Member Functions**

- quint64 **pageSize** ()
- quint64 **totalMemory** ()
- quint64 **freeMemory** ()
- QString **stackTrace** ()
- QString **platformName** ()

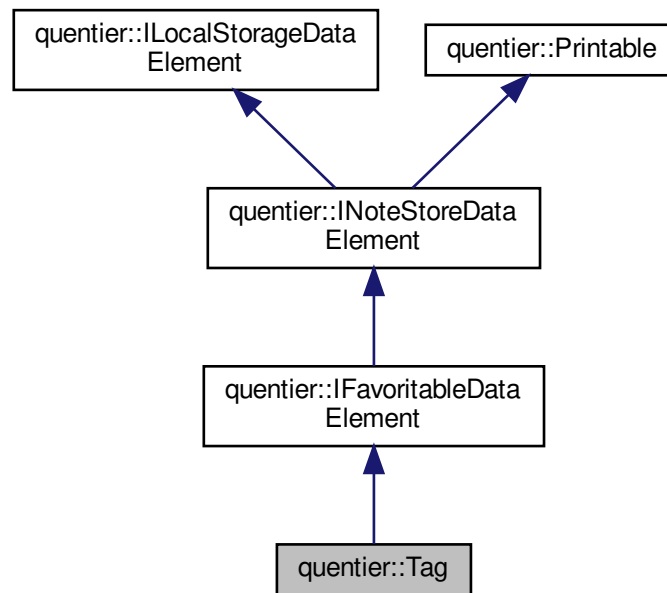
## 4.71 quantier::Tag Class Reference

Inheritance diagram for quantier::Tag:





Collaboration diagram for quantier::Tag:



## Public Member Functions

- **Tag** (const [Tag](#) &other)
- **Tag** ([Tag](#) &&other)
- [Tag](#) & **operator=** (const [Tag](#) &other)
- [Tag](#) & **operator=** ([Tag](#) &&other)
- **Tag** (const qevercloud::Tag &other)
- **Tag** (qevercloud::Tag &&other)
- bool **operator==** (const [Tag](#) &other) const
- bool **operator!=** (const [Tag](#) &other) const
- const qevercloud::Tag & **qevercloudTag** () const
- qevercloud::Tag & **qevercloudTag** ()
- virtual void **clear** () override
- virtual bool **hasGuid** () const override
- virtual const QString & **guid** () const override
- virtual void **setGuid** (const QString &guid) override
- virtual bool **hasUpdateSequenceNumber** () const override
- virtual qint32 **updateSequenceNumber** () const override
- virtual void **setUpdateSequenceNumber** (const qint32 usn) override
- virtual bool **checkParameters** ([ErrorString](#) &errorDescription) const override
- bool **hasName** () const
- const QString & **name** () const
- void **setName** (const QString &name)
- bool **hasParentGuid** () const
- const QString & **parentGuid** () const
- void **setParentGuid** (const QString &parentGuid)

- bool **hasParentLocalUid** () const
- const QString & **parentLocalUid** () const
- void **setParentLocalUid** (const QString &parentLocalUid)
- bool **hasLinkedNotebookGuid** () const
- const QString & **linkedNotebookGuid** () const
- void **setLinkedNotebookGuid** (const QString &linkedNotebookGuid)
- virtual QTextStream & **print** (QTextStream &strm) const override

### Static Public Member Functions

- static bool **validateName** (const QString &name, [ErrorString](#) \*pErrorDescription=nullptr)

### Additional Inherited Members

## 4.72 quotient::ResourceRecognitionIndexItem::TextItem Struct Reference

### Public Member Functions

- bool **operator==** (const [TextItem](#) &other) const

### Public Attributes

- QString **m\_text**
- int **m\_weight** = -1

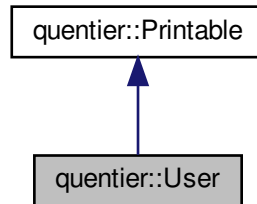
## 4.73 quotient::UidGenerator Class Reference

### Static Public Member Functions

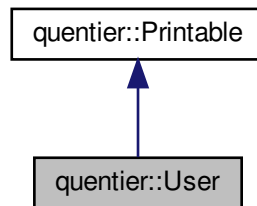
- static QString **Generate** ()
- static QString **UidToString** (const QUuid &uid)

## 4.74 quantier::User Class Reference

Inheritance diagram for quantier::User:



Collaboration diagram for quantier::User:



### Public Types

- using **PrivilegeLevel** = qevercloud::PrivilegeLevel
- using **ServiceLevel** = qevercloud::ServiceLevel

### Public Member Functions

- **User** (const qevercloud::User &user)
- **User** (qevercloud::User &&user)
- **User** (const **User** &other)
- **User** (**User** &&other)
- **User** & **operator=** (const **User** &other)
- **User** & **operator=** (**User** &&other)
- bool **operator==** (const **User** &other) const
- bool **operator!=** (const **User** &other) const
- const qevercloud::User & **qevercloudUser** () const
- qevercloud::User & **qevercloudUser** ()

- void **clear** ()
- bool **isDirty** () const
- void **setDirty** (const bool dirty)
- bool **isLocal** () const
- void **setLocal** (const bool local)
- bool **checkParameters** ([ErrorString](#) &errorDescription) const
- bool **hasId** () const
- quint32 **id** () const
- void **setId** (const quint32 id)
- bool **hasUsername** () const
- const QString & **username** () const
- void **setUsername** (const QString &username)
- bool **hasEmail** () const
- const QString & **email** () const
- void **setEmail** (const QString &email)
- bool **hasName** () const
- const QString & **name** () const
- void **setName** (const QString &name)
- bool **hasTimezone** () const
- const QString & **timezone** () const
- void **setTimezone** (const QString &timezone)
- bool **hasPrivilegeLevel** () const
- PrivilegeLevel **privilegeLevel** () const
- void **setPrivilegeLevel** (const quint8 level)
- bool **hasServiceLevel** () const
- ServiceLevel **serviceLevel** () const
- void **setServiceLevel** (const quint8 level)
- bool **hasCreationTimestamp** () const
- quint64 **creationTimestamp** () const
- void **setCreationTimestamp** (const quint64 timestamp)
- bool **hasModificationTimestamp** () const
- quint64 **modificationTimestamp** () const
- void **setModificationTimestamp** (const quint64 timestamp)
- bool **hasDeletionTimestamp** () const
- quint64 **deletionTimestamp** () const
- void **setDeletionTimestamp** (const quint64 timestamp)
- bool **hasActive** () const
- bool **active** () const
- void **setActive** (const bool active)
- bool **hasShardId** () const
- const QString & **shardId** () const
- void **setShardId** (const QString &shardId)
- bool **hasUserAttributes** () const
- const qevercloud::UserAttributes & **userAttributes** () const
- void **setUserAttributes** (qevercloud::UserAttributes &&attributes)
- bool **hasAccounting** () const
- const qevercloud::Accounting & **accounting** () const
- void **setAccounting** (qevercloud::Accounting &&accounting)
- bool **hasBusinessUserInfo** () const
- const qevercloud::BusinessUserInfo & **businessUserInfo** () const
- void **setBusinessUserInfo** (qevercloud::BusinessUserInfo &&info)
- bool **hasPhotoUrl** () const
- QString **photoUrl** () const
- void **setPhotoUrl** (const QString &photoUrl)
- bool **hasPhotoLastUpdateTimestamp** () const

- `qint64 photoLastUpdateTimestamp () const`
- `void setPhotoLastUpdateTimestamp (const qint64 timestamp)`
- `bool hasAccountLimits () const`
- `const qevercloud::AccountLimits & accountLimits () const`
- `void setAccountLimits (qevercloud::AccountLimits &&limits)`
- `virtual QTextStream & print (QTextStream &strm) const` override

## Friends

- class `Notebook`

## Additional Inherited Members



# Index

- ~ApplicationSettings
  - quentier::ApplicationSettings, [15](#)
- AccessDenied
  - quentier::IKeychainService, [53](#)
- AccessDeniedByUser
  - quentier::IKeychainService, [53](#)
- accountHighUsn
  - quentier::LocalStorageManager, [103](#)
- active
  - quentier::SynchronizationManager, [195](#)
- addEnResource
  - quentier::LocalStorageManager, [104](#)
- addLinkedNotebook
  - quentier::LocalStorageManager, [104](#)
- addNote
  - quentier::LocalStorageManager, [105](#)
- addNotebook
  - quentier::LocalStorageManager, [105](#)
- addSavedSearch
  - quentier::LocalStorageManager, [106](#)
- addTag
  - quentier::LocalStorageManager, [106](#)
- addUser
  - quentier::LocalStorageManager, [106](#)
- ApplicationSettings
  - quentier::ApplicationSettings, [14](#), [15](#)
- apply
  - quentier::ILocalStoragePatch, [62](#)
- authenticate
  - quentier::SynchronizationManager, [195](#)
- authenticateCurrentAccount
  - quentier::SynchronizationManager, [195](#)
- authenticateToSharedNotebook
  - quentier::INoteStore, [71](#)
- authenticationFinished
  - quentier::SynchronizationManager, [195](#)
- authenticationRevoked
  - quentier::SynchronizationManager, [196](#)
- backend
  - quentier::NoteEditor, [157](#)
- backupLocalStorage
  - quentier::ILocalStoragePatch, [62](#)
- backupProgress
  - quentier::ILocalStoragePatch, [63](#)
- beginGroup
  - quentier::ApplicationSettings, [15](#), [16](#)
- beginReadArray
  - quentier::ApplicationSettings, [16](#)
- beginWriteArray
  - quentier::ApplicationSettings, [17](#)
- checkLinkedNotebooks
  - quentier::DefaultLocalStorageCacheExpiryChecker, [32](#)
  - quentier::ILocalStorageCacheExpiryChecker, [59](#)
- checkNotebooks
  - quentier::DefaultLocalStorageCacheExpiryChecker, [32](#)
  - quentier::ILocalStorageCacheExpiryChecker, [59](#)
- checkNotes
  - quentier::DefaultLocalStorageCacheExpiryChecker, [32](#)
  - quentier::ILocalStorageCacheExpiryChecker, [59](#)
- checkResources
  - quentier::DefaultLocalStorageCacheExpiryChecker, [32](#)
  - quentier::ILocalStorageCacheExpiryChecker, [59](#)
- checkSavedSearches
  - quentier::DefaultLocalStorageCacheExpiryChecker, [33](#)
  - quentier::ILocalStorageCacheExpiryChecker, [59](#)
- checkTags
  - quentier::DefaultLocalStorageCacheExpiryChecker, [33](#)
  - quentier::ILocalStorageCacheExpiryChecker, [60](#)
- checkVersion
  - quentier::IUserStore, [87](#)
- cleanupExternalHtml
  - quentier::ENMLConverter, [37](#)
- clear
  - quentier::NoteEditor, [158](#)
- ClearDatabase
  - quentier::LocalStorageManager, [102](#)
- clone
  - quentier::DefaultLocalStorageCacheExpiryChecker, [33](#)
  - quentier::ILocalStorageCacheExpiryChecker, [60](#)
- contains
  - quentier::ApplicationSettings, [17](#), [18](#)
- convertToNote
  - quentier::NoteEditor, [158](#)
- CouldNotDeleteEntry
  - quentier::IKeychainService, [53](#)
- createNote
  - quentier::INoteStore, [71](#)
- createNotebook
  - quentier::INoteStore, [72](#)
- createSavedSearch

- quentier::INoteStore, [72](#)
- createTag
  - quentier::INoteStore, [73](#)
- currentNoteLocalUid
  - quentier::NoteEditor, [158](#)
- defaultFont
  - quentier::NoteEditor, [158](#)
- defaultPalette
  - quentier::NoteEditor, [158](#)
- defaultShortcut
  - quentier::ShortcutManager, [187](#), [188](#)
- deletePasswordJobFinished
  - quentier::IKeychainService, [53](#)
- deleteUser
  - quentier::LocalStorageManager, [107](#)
- detectedConflictDuringLocalChangesSending
  - quentier::SynchronizationManager, [196](#)
- displayName
  - quentier::Account, [11](#)
- downloadNoteThumbnailsOption
  - quentier::SynchronizationManager, [196](#)
- enResourceCount
  - quentier::LocalStorageManager, [107](#)
- EntryNotFound
  - quentier::IKeychainService, [53](#)
- ErrorCode
  - quentier::IKeychainService, [53](#)
- evernoteAccountType
  - quentier::Account, [11](#)
- evernoteHost
  - quentier::Account, [11](#)
- exportNotesToEnex
  - quentier::ENMLConverter, [38](#)
- expungeEnResource
  - quentier::LocalStorageManager, [108](#)
- expungeLinkedNotebook
  - quentier::LocalStorageManager, [108](#)
- expungeNote
  - quentier::LocalStorageManager, [109](#)
- expungeNotebook
  - quentier::LocalStorageManager, [109](#)
- expungeNotelessTagsFromLinkedNotebooks
  - quentier::LocalStorageManager, [110](#)
- expungeSavedSearch
  - quentier::LocalStorageManager, [110](#)
- expungeTag
  - quentier::LocalStorageManager, [110](#)
- expungeUser
  - quentier::LocalStorageManager, [111](#)
- failed
  - quentier::SynchronizationManager, [196](#)
- findDefaultNotebook
  - quentier::LocalStorageManager, [111](#)
- findDefaultOrLastUsedNotebook
  - quentier::LocalStorageManager, [112](#)
- findEnResource
  - quentier::LocalStorageManager, [112](#)
- findLastUsedNotebook
  - quentier::LocalStorageManager, [113](#)
- findLinkedNotebook
  - quentier::LocalStorageManager, [113](#)
- findNote
  - quentier::LocalStorageManager, [113](#)
- findNotebook
  - quentier::LocalStorageManager, [114](#)
- findNoteLocalUidsWithSearchQuery
  - quentier::LocalStorageManager, [114](#)
- findNotesWithSearchQuery
  - quentier::LocalStorageManager, [115](#)
- findSavedSearch
  - quentier::LocalStorageManager, [115](#)
- findTag
  - quentier::LocalStorageManager, [116](#)
- findUser
  - quentier::LocalStorageManager, [116](#)
- finished
  - quentier::SynchronizationManager, [197](#)
- fromVersion
  - quentier::ILocalStoragePatch, [63](#)
- getAccountLimits
  - quentier::IUserStore, [88](#)
- getLinkedNotebookSyncChunk
  - quentier::INoteStore, [73](#)
- getLinkedNotebookSyncState
  - quentier::INoteStore, [74](#)
- getNote
  - quentier::INoteStore, [75](#)
- getNoteAsync
  - quentier::INoteStore, [75](#)
- GetNoteOption
  - quentier::LocalStorageManager, [101](#)
- getResource
  - quentier::INoteStore, [76](#)
- getResourceAsync
  - quentier::INoteStore, [77](#)
- GetResourceOption
  - quentier::LocalStorageManager, [102](#)
- getSyncChunk
  - quentier::INoteStore, [77](#)
- getSyncState
  - quentier::INoteStore, [78](#)
- getUser
  - quentier::IUserStore, [88](#)
- highestSupportedLocalStorageVersion
  - quentier::LocalStorageManager, [117](#)
- htmlToQTextDocument
  - quentier::ENMLConverter, [38](#)
- id
  - quentier::Account, [11](#)
- idleTime
  - quentier::NoteEditor, [158](#)
- importEnex



- quentier::ENMLConverter, 39
- inAppNoteLinkPasteRequested
  - quentier::NoteEditor, 159
- IncludeMilliseconds
  - quentier::DateTimePrint, 29
- IncludeNumericTimestamp
  - quentier::DateTimePrint, 29
- IncludeTimezone
  - quentier::DateTimePrint, 29
- initialize
  - quentier::NoteEditor, 159
- isEditorPageModified
  - quentier::NoteEditor, 159
- isEmpty
  - quentier::Account, 11
- isLocalStorageVersionTooHigh
  - quentier::LocalStorageManager, 117
- isModified
  - quentier::NoteEditor, 160
- isNoteLoaded
  - quentier::NoteEditor, 160
- linkedNotebookCount
  - quentier::LocalStorageManager, 117
- linkedNotebooksNotesDownloadProgress
  - quentier::SynchronizationManager, 197
- linkedNotebooksResourcesDownloadProgress
  - quentier::SynchronizationManager, 197
- linkedNotebooksSyncChunksDownloaded
  - quentier::SynchronizationManager, 198
- linkedNotebookSyncChunksDownloadProgress
  - quentier::SynchronizationManager, 198
- listAllLinkedNotebooks
  - quentier::LocalStorageManager, 118
- listAllNotebooks
  - quentier::LocalStorageManager, 118
- listAllSavedSearches
  - quentier::LocalStorageManager, 119
- listAllSharedNotebooks
  - quentier::LocalStorageManager, 120
- listAllTags
  - quentier::LocalStorageManager, 120
- listAllTagsPerNote
  - quentier::LocalStorageManager, 121
- listLinkedNotebooks
  - quentier::LocalStorageManager, 121
- listNotebooks
  - quentier::LocalStorageManager, 122
- listNotes
  - quentier::LocalStorageManager, 123
- listNotesByLocalUids
  - quentier::LocalStorageManager, 123
- listNotesPerNotebook
  - quentier::LocalStorageManager, 124
- listNotesPerNotebooksAndTags
  - quentier::LocalStorageManager, 125
- listNotesPerTag
  - quentier::LocalStorageManager, 125
- ListObjectsOption
  - quentier::LocalStorageManager, 102
- listSavedSearches
  - quentier::LocalStorageManager, 126
- listSharedNotebooksPerNotebookGuid
  - quentier::LocalStorageManager, 127
- listTags
  - quentier::LocalStorageManager, 127
- listTagsWithNoteLocalUids
  - quentier::LocalStorageManager, 128
- LocalStorageManager
  - quentier::LocalStorageManager, 103
- localStorageRequiresUpgrade
  - quentier::LocalStorageManager, 129
- localStorageVersion
  - quentier::LocalStorageManager, 129
- name
  - quentier::Account, 12
- NoBackendAvailable
  - quentier::IKeychainService, 53
- NoError
  - quentier::IKeychainService, 53
- notebookCount
  - quentier::LocalStorageManager, 130
- notebookModifier
  - quentier::NoteSearchQuery, 168
- noteCount
  - quentier::LocalStorageManager, 130
- noteCountPerNotebook
  - quentier::LocalStorageManager, 130
- noteCountPerNotebooksAndTags
  - quentier::LocalStorageManager, 131
- noteCountPerTag
  - quentier::LocalStorageManager, 131
- noteCountsPerAllTags
  - quentier::LocalStorageManager, 132
- notesDownloadProgress
  - quentier::SynchronizationManager, 198
- noteStoreUrl
  - quentier::INoteStore, 79
- notifySyncStateUpdated
  - quentier::ISyncStateStorage, 86
- NotImplemented
  - quentier::IKeychainService, 53
- onReadFileRequest
  - quentier::FileIOProcessorAsync, 45
- onWriteFileRequest
  - quentier::FileIOProcessorAsync, 45
- Option
  - quentier::DateTimePrint, 29
- OtherError
  - quentier::IKeychainService, 53
- OverrideLock
  - quentier::LocalStorageManager, 102
- patchLongDescription
  - quentier::ILocalStoragePatch, 63
- patchShortDescription

- quentier::ILocalStoragePatch, 63
- preparedDirtyObjectsForSending
  - quentier::SynchronizationManager, 199
- preparedLinkedNotebooksDirtyObjectsForSending
  - quentier::SynchronizationManager, 199
- progress
  - quentier::ILocalStoragePatch, 64
- quentier::Account, 9
  - displayName, 11
  - evernoteAccountType, 11
  - evernoteHost, 11
  - id, 11
  - isEmpty, 11
  - name, 12
  - setDisplayName, 12
  - shardId, 12
  - type, 12
- quentier::ApplicationSettings, 13
  - ~ApplicationSettings, 15
  - ApplicationSettings, 14, 15
  - beginGroup, 15, 16
  - beginReadArray, 16
  - beginWriteArray, 17
  - contains, 17, 18
  - remove, 18, 19
  - setValue, 19
  - value, 20
- quentier::ApplicationSettings::ArrayCloser, 22
- quentier::ApplicationSettings::GroupCloser, 48
- quentier::ApplicationSettingsInitializationException, 21
- quentier::AuthenticationManager, 23
- quentier::DatabaseLockedException, 24
- quentier::DatabaseLockFailedException, 25
- quentier::DatabaseOpeningException, 26
- quentier::DatabaseRequestException, 27
- quentier::DateTimePrint, 29
  - IncludeMilliseconds, 29
  - IncludeNumericTimestamp, 29
  - IncludeTimezone, 29
  - Option, 29
- quentier::DecryptedTextManager, 29
- quentier::DefaultLocalStorageCacheExpiryChecker, 30
  - checkLinkedNotebooks, 32
  - checkNotebooks, 32
  - checkNotes, 32
  - checkResources, 32
  - checkSavedSearches, 33
  - checkTags, 33
  - clone, 33
- quentier::EmptyDataElementException, 34
- quentier::EncryptionManager, 35
- quentier::ENMLConverter, 36
  - cleanupExternalHtml, 37
  - exportNotesToEnex, 38
  - htmlToQTextDocument, 38
  - importEnex, 39
- quentier::ENMLConverter::NoteContentToHtmlExtraData, 154
- quentier::ENMLConverter::SkipHtmlElementRule, 189
- quentier::ErrorString, 39
- quentier::EventLoopWithExitStatus, 41
- quentier::FileCopier, 42
- quentier::FileIOProcessorAsync, 44
  - onReadFileRequest, 45
  - onWriteFileRequest, 45
  - readFileRequestProcessed, 46
  - setIdleTimePeriod, 46
  - writeFileRequestProcessed, 46
- quentier::FileSystemWatcher, 47
- quentier::HTMLCleaner, 49
- quentier::IAuthenticationManager, 49
- quentier::IFavoritableDataElement, 50
- quentier::IKeychainService, 51
  - AccessDenied, 53
  - AccessDeniedByUser, 53
  - CouldNotDeleteEntry, 53
  - deletePasswordJobFinished, 53
  - EntryNotFound, 53
  - ErrorCode, 53
  - NoBackendAvailable, 53
  - NoError, 53
  - NotImplemented, 53
  - OtherError, 53
  - readPasswordJobFinished, 55
  - startDeletePasswordJob, 55
  - startReadPasswordJob, 55
  - startWritePasswordJob, 56
  - writePasswordJobFinished, 56
- quentier::ILocalStorageCacheExpiryChecker, 57
  - checkLinkedNotebooks, 59
  - checkNotebooks, 59
  - checkNotes, 59
  - checkResources, 59
  - checkSavedSearches, 59
  - checkTags, 60
  - clone, 60
- quentier::ILocalStorageDataElement, 60
- quentier::ILocalStoragePatch, 61
  - apply, 62
  - backupLocalStorage, 62
  - backupProgress, 63
  - fromVersion, 63
  - patchLongDescription, 63
  - patchShortDescription, 63
  - progress, 64
  - removeLocalStorageBackup, 64
  - restoreBackupProgress, 64
  - restoreLocalStorageFromBackup, 65
  - toVersion, 65
- quentier::INoteEditorBackend, 66
- quentier::INoteStore, 69
  - authenticateToSharedNotebook, 71
  - createNote, 71
  - createNotebook, 72
  - createSavedSearch, 72
  - createTag, 73

- getLinkedNotebookSyncChunk, 73
- getLinkedNotebookSyncState, 74
- getNote, 75
- getNoteAsync, 75
- getResource, 76
- getResourceAsync, 77
- getSyncChunk, 77
- getSyncState, 78
- noteStoreUrl, 79
- setAuthData, 79
- setNoteStoreUrl, 79
- stop, 79
- updateNote, 79
- updateNotebook, 80
- updateSavedSearch, 80
- updateTag, 81
- quentier::INoteStoreDataElement, 81
- quentier::IQuentierException, 82
- quentier::ISyncStateStorage, 85
  - notifySyncStateUpdated, 86
- quentier::ISyncStateStorage::ISyncState, 84
- quentier::IUserStore, 87
  - checkVersion, 87
  - getAccountLimits, 88
  - getUser, 88
  - setAuthData, 89
- quentier::LimitedStack< T >, 89
- quentier::LinkedNotebook, 90
- quentier::LocalStorageCacheManager, 92
- quentier::LocalStorageCacheManagerException, 94
- quentier::LocalStorageManager, 95
  - accountHighUsn, 103
  - addEnResource, 104
  - addLinkedNotebook, 104
  - addNote, 105
  - addNotebook, 105
  - addSavedSearch, 106
  - addTag, 106
  - addUser, 106
  - ClearDatabase, 102
  - deleteUser, 107
  - enResourceCount, 107
  - expungeEnResource, 108
  - expungeLinkedNotebook, 108
  - expungeNote, 109
  - expungeNotebook, 109
  - expungeNotelessTagsFromLinkedNotebooks, 110
  - expungeSavedSearch, 110
  - expungeTag, 110
  - expungeUser, 111
  - findDefaultNotebook, 111
  - findDefaultOrLastUsedNotebook, 112
  - findEnResource, 112
  - findLastUsedNotebook, 113
  - findLinkedNotebook, 113
  - findNote, 113
  - findNotebook, 114
  - findNoteLocalUidsWithSearchQuery, 114
  - findNotesWithSearchQuery, 115
  - findSavedSearch, 115
  - findTag, 116
  - findUser, 116
  - GetNoteOption, 101
  - GetResourceOption, 102
  - highestSupportedLocalStorageVersion, 117
  - isLocalStorageVersionTooHigh, 117
  - linkedNotebookCount, 117
  - listAllLinkedNotebooks, 118
  - listAllNotebooks, 118
  - listAllSavedSearches, 119
  - listAllSharedNotebooks, 120
  - listAllTags, 120
  - listAllTagsPerNote, 121
  - listLinkedNotebooks, 121
  - listNotebooks, 122
  - listNotes, 123
  - listNotesByLocalUids, 123
  - listNotesPerNotebook, 124
  - listNotesPerNotebooksAndTags, 125
  - listNotesPerTag, 125
  - ListObjectsOption, 102
  - listSavedSearches, 126
  - listSharedNotebooksPerNotebookGuid, 127
  - listTags, 127
  - listTagsWithNoteLocalUids, 128
  - LocalStorageManager, 103
  - localStorageRequiresUpgrade, 129
  - localStorageVersion, 129
  - notebookCount, 130
  - noteCount, 130
  - noteCountPerNotebook, 130
  - noteCountPerNotebooksAndTags, 131
  - noteCountPerTag, 131
  - noteCountsPerAllTags, 132
  - OverrideLock, 102
  - requiredLocalStoragePatches, 132
  - savedSearchCount, 132
  - StartupOption, 102
  - switchUser, 133
  - tagCount, 133
  - updateEnResource, 134
  - updateLinkedNotebook, 134
  - updateNote, 134
  - updateNotebook, 135
  - UpdateNoteOption, 102
  - UpdateResourceBinaryData, 103
  - UpdateResourceMetadata, 103
  - updateSavedSearch, 136
  - updateTag, 136
  - UpdateTags, 103
  - updateUser, 137
  - upgradeProgress, 137
  - userCount, 138
  - WithBinaryData, 102
  - WithResourceBinaryData, 101
  - WithResourceMetadata, 101

- quentier::LocalStorageManagerAsync, 138
- quentier::LoggerInitializationException, 145
- quentier::LRUCache< Key, Value, Allocator >, 146
- quentier::Note, 147
- quentier::Notebook, 150
- quentier::NoteEditor, 154
  - backend, 157
  - clear, 158
  - convertToNote, 158
  - currentNoteLocalUid, 158
  - defaultFont, 158
  - defaultPalette, 158
  - idleTime, 158
  - inAppNoteLinkPasteRequested, 159
  - initialize, 159
  - isEditorPageModified, 159
  - isModified, 160
  - isNoteLoaded, 160
  - saveNoteToLocalStorage, 160
  - setAccount, 160
  - setBackend, 160
  - setCurrentNoteLocalUid, 160
  - setDefaultFont, 161
  - setDefaultPalette, 161
  - setFocus, 161
  - setInitialPageHtml, 161
  - setNoteDeletedPageHtml, 162
  - setNoteLoadingPageHtml, 162
  - setNoteNotFoundPageHtml, 162
  - setNoteTitle, 162
  - setTagIds, 162
  - setUndoStack, 163
  - undoStack, 163
- quentier::NoteEditorInitializationException, 163
- quentier::NoteEditorPluginInitializationException, 164
- quentier::NoteSearchQuery, 165
  - notebookModifier, 168
  - queryString, 168
- quentier::NullPtrException, 168
- quentier::Printable, 170
- quentier::QuentierApplication, 171
- quentier::QuentierUndoCommand, 172
- quentier::Resource, 173
- quentier::ResourceRecognitionIndexItem, 176
- quentier::ResourceRecognitionIndexItem::BarcodeItem, 24
- quentier::ResourceRecognitionIndexItem::ObjectItem, 169
- quentier::ResourceRecognitionIndexItem::ShapeItem, 181
- quentier::ResourceRecognitionIndexItem::TextItem, 206
- quentier::ResourceRecognitionIndices, 178
- quentier::SavedSearch, 179
- quentier::SharedNote, 182
- quentier::SharedNotebook, 184
- quentier::ShortcutManager, 186
  - defaultShortcut, 187, 188
  - shortcut, 188
  - userShortcut, 188, 189
- quentier::SpellChecker, 191
- quentier::StringUtils, 192
- quentier::StringUtils::StringFilterPredicate, 192
- quentier::SynchronizationManager, 192
  - active, 195
  - authenticate, 195
  - authenticateCurrentAccount, 195
  - authenticationFinished, 195
  - authenticationRevoked, 196
  - detectedConflictDuringLocalChangesSending, 196
  - downloadNoteThumbnailsOption, 196
  - failed, 196
  - finished, 197
  - linkedNotebooksNotesDownloadProgress, 197
  - linkedNotebooksResourcesDownloadProgress, 197
  - linkedNotebooksSyncChunksDownloaded, 198
  - linkedNotebookSyncChunksDownloadProgress, 198
  - notesDownloadProgress, 198
  - preparedDirtyObjectsForSending, 199
  - preparedLinkedNotebooksDirtyObjectsForSending, 199
  - rateLimitExceeded, 199
  - remoteToLocalSyncDone, 199
  - remoteToLocalSyncStopped, 200
  - resourcesDownloadProgress, 200
  - revokeAuthentication, 200
  - sendLocalChangesStopped, 200
  - setAccount, 200
  - setAccountDone, 201
  - setDownloadInkNoteImages, 201
  - setDownloadInkNoteImagesDone, 201
  - setDownloadNoteThumbnails, 201
  - setDownloadNoteThumbnailsDone, 201
  - setInkNoteImagesStoragePath, 202
  - setInkNoteImagesStoragePathDone, 202
  - started, 202
  - stop, 202
  - stopped, 202
  - syncChunksDownloaded, 202
  - syncChunksDownloadProgress, 203
  - SynchronizationManager, 194
  - synchronize, 203
  - willRepeatRemoteToLocalSyncAfterSendingChanges, 203
- quentier::SysInfo, 203
- quentier::Tag, 204
- quentier::UidGenerator, 206
- quentier::User, 207
- queryString
  - quentier::NoteSearchQuery, 168
- rateLimitExceeded
  - quentier::SynchronizationManager, 199
- readFileRequestProcessed
  - quentier::FileIOProcessorAsync, 46
- readPasswordJobFinished

- quantier::IKeychainService, 55
- remoteToLocalSyncDone
  - quantier::SynchronizationManager, 199
- remoteToLocalSyncStopped
  - quantier::SynchronizationManager, 200
- remove
  - quantier::ApplicationSettings, 18, 19
- removeLocalStorageBackup
  - quantier::ILocalStoragePatch, 64
- requiredLocalStoragePatches
  - quantier::LocalStorageManager, 132
- resourcesDownloadProgress
  - quantier::SynchronizationManager, 200
- restoreBackupProgress
  - quantier::ILocalStoragePatch, 64
- restoreLocalStorageFromBackup
  - quantier::ILocalStoragePatch, 65
- revokeAuthentication
  - quantier::SynchronizationManager, 200
- savedSearchCount
  - quantier::LocalStorageManager, 132
- saveNoteToLocalStorage
  - quantier::NoteEditor, 160
- sendLocalChangesStopped
  - quantier::SynchronizationManager, 200
- setAccount
  - quantier::NoteEditor, 160
  - quantier::SynchronizationManager, 200
- setAccountDone
  - quantier::SynchronizationManager, 201
- setAuthData
  - quantier::INoteStore, 79
  - quantier::IUserStore, 89
- setBackend
  - quantier::NoteEditor, 160
- setCurrentNoteLocalUid
  - quantier::NoteEditor, 160
- setDefaultFont
  - quantier::NoteEditor, 161
- setDefaultPalette
  - quantier::NoteEditor, 161
- setDisplayName
  - quantier::Account, 12
- setDownloadInkNoteImages
  - quantier::SynchronizationManager, 201
- setDownloadInkNoteImagesDone
  - quantier::SynchronizationManager, 201
- setDownloadNoteThumbnails
  - quantier::SynchronizationManager, 201
- setDownloadNoteThumbnailsDone
  - quantier::SynchronizationManager, 201
- setFocus
  - quantier::NoteEditor, 161
- setIdleTimePeriod
  - quantier::FileIOProcessorAsync, 46
- setInitialPageHtml
  - quantier::NoteEditor, 161
- setInkNoteImagesStoragePath
  - quantier::SynchronizationManager, 202
- setInkNoteImagesStoragePathDone
  - quantier::SynchronizationManager, 202
- setNoteDeletedPageHtml
  - quantier::NoteEditor, 162
- setNoteLoadingPageHtml
  - quantier::NoteEditor, 162
- setNoteNotFoundPageHtml
  - quantier::NoteEditor, 162
- setNoteStoreUrl
  - quantier::INoteStore, 79
- setNoteTitle
  - quantier::NoteEditor, 162
- setTagIds
  - quantier::NoteEditor, 162
- setUndoStack
  - quantier::NoteEditor, 163
- setValue
  - quantier::ApplicationSettings, 19
- shardId
  - quantier::Account, 12
- shortcut
  - quantier::ShortcutManager, 188
- startDeletePasswordJob
  - quantier::IKeychainService, 55
- started
  - quantier::SynchronizationManager, 202
- startReadPasswordJob
  - quantier::IKeychainService, 55
- StartupOption
  - quantier::LocalStorageManager, 102
- startWritePasswordJob
  - quantier::IKeychainService, 56
- stop
  - quantier::INoteStore, 79
  - quantier::SynchronizationManager, 202
- stopped
  - quantier::SynchronizationManager, 202
- switchUser
  - quantier::LocalStorageManager, 133
- syncChunksDownloaded
  - quantier::SynchronizationManager, 202
- syncChunksDownloadProgress
  - quantier::SynchronizationManager, 203
- SynchronizationManager
  - quantier::SynchronizationManager, 194
- synchronize
  - quantier::SynchronizationManager, 203
- tagCount
  - quantier::LocalStorageManager, 133
- toVersion
  - quantier::ILocalStoragePatch, 65
- type
  - quantier::Account, 12
- undoStack
  - quantier::NoteEditor, 163
- updateEnResource

- quentier::LocalStorageManager, 134
- updateLinkedNotebook
  - quentier::LocalStorageManager, 134
- updateNote
  - quentier::INoteStore, 79
  - quentier::LocalStorageManager, 134
- updateNotebook
  - quentier::INoteStore, 80
  - quentier::LocalStorageManager, 135
- UpdateNoteOption
  - quentier::LocalStorageManager, 102
- UpdateResourceBinaryData
  - quentier::LocalStorageManager, 103
- UpdateResourceMetadata
  - quentier::LocalStorageManager, 103
- updateSavedSearch
  - quentier::INoteStore, 80
  - quentier::LocalStorageManager, 136
- updateTag
  - quentier::INoteStore, 81
  - quentier::LocalStorageManager, 136
- UpdateTags
  - quentier::LocalStorageManager, 103
- updateUser
  - quentier::LocalStorageManager, 137
- upgradeProgress
  - quentier::LocalStorageManager, 137
- userCount
  - quentier::LocalStorageManager, 138
- userShortcut
  - quentier::ShortcutManager, 188, 189
- value
  - quentier::ApplicationSettings, 20
- willRepeatRemoteToLocalSyncAfterSendingChanges
  - quentier::SynchronizationManager, 203
- WithBinaryData
  - quentier::LocalStorageManager, 102
- WithResourceBinaryData
  - quentier::LocalStorageManager, 101
- WithResourceMetadata
  - quentier::LocalStorageManager, 101
- writeFileRequestProcessed
  - quentier::FileIOProcessorAsync, 46
- writePasswordJobFinished
  - quentier::IKeychainService, 56